МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Черкаський національний університет
імені Богдана Хмельницького
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
Bohdan Khmelnytsky
National University of Cherkasy

ISSN 2076-5886 (Print)

ВІСНИК ЧЕРКАСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

Серія
ПРИКЛАДНА МАТЕМАТИКА. ІНФОРМАТИКА

BULLETIN OF THE CHERKASY BOHDAN KHMELNYTSKY NATIONAL UNIVERSITY

APPLIED MATHEMATICS. INFORMATICS

ВИПУСК 1
ISSUE 1

Черкаси, 2024
Cherkasy, 2024

Засновник, редакція, видавець і виготовлювач –
Черкаський національний університет імені Богдана Хмельницького.
Свідоцтво про державну реєстрацію КВ № 16161-4633 ПР від 11.12.2009.
Свідоцтво про державну реєстрацію КВ № 23973-13813 Р від 21.05.2019.

Журнал розрахований на математиків, спеціалістів у галузі ІТ, викладачів, науковців, аспірантів, студентів.

Випуск № 1 наукового журналу «Вісник Черкаського національного університету імені Богдана Хмельницького. Серія «Прикладна математика. Інформатика» рекомендовано до друку та до поширення через мережу Інтернет Вченою радою Черкаського національного університету імені Богдана Хмельницького (протокол № 4 від 19.12.2024 року).

Журнал індексується Google Scholar.

За дотримання права інтелектуальної власності, достовірність матеріалів та обґрунтування висновків відповідають автори.

Адреса редакційної колегії:
18031, Черкаси, бул. Шевченка, 79
Черкаський національний університет імені Богдана Хмельницького, корпус № 3, к. 307
e-mail: ami.cdu@ukr.net

З електронною версією журналу можна ознайомитися за адресою: http://ami-ejournal.cdu.edu.ua/

Founder, editorial, publisher and manufacturer –
Bohdan Khmelnytsky National University of Cherkasy.
State registration certificate: KV No. 16161-4633 PR dated 11.12.2009.
State registration certificate: KV No. 23973-13813 R dated 21.05.2019.

This journal is meant for mathematicians, IT specialists, teachers, researchers, postgraduates and students.

Issue № 1 of the scientific journal «Bulletin of the Cherkasy Bohdan Khmelnytsky national university. Applied mathematics. Informatics» is recommended for publication and dissemination through the Internet by the Academic Council of Bohdan Khmelnytsky National University of Cherkasy (protocol number 4 dated 19.12.2024).

The journal is indexed in Google Scholar.

The authors are responsible for the observance of the intellectual property right, for the reliability of the materials and for the substantiation of the conclusions.

Editorial office address:
18031, Cherkasy, Shevchenko Blvd., 79
Bohdan Khmelnytsky National University of Cherkasy, building 3, ap. 307
e-mail: ami.cdu@ukr.net

All electronic versions of articles are available on the website edition http://ami-ejournal.cdu.edu.ua/

СЕКЦІЯ «ПРИКЛАДНА МАТЕМАТИКА»

УДК 519.6:004.421

DOI 10.31651/2076-5886-2024-1-4-21

PACS 02.10.Ox, 02.70.Hm, 07.05.Tp

ГЛАДКА Людмила Іванівна
к.ф.-м.н., доцент кафедри автоматизації та комп'ютерно-інтегрованих технологій, Черкаський національний університет імені Б. Хмельницького
e-mail: l_i_gladka@vu.cdu.edu.ua
ORCID 0000-0002-7030-9666

ЧАЛИЙ Антон Анатолійович
розробник ПЗ, ТОВ «ЮніСтар», м. Сміла
e-mail: anton.chaliy@gmail.com

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ПОШУКУ МАРШРУТУ НА ЛАБІРИНТАХ ТА ДВОВИМІРНИХ СІТКОВИХ СТРУКТУРАХ

У статті проведено порівняльний аналіз швидкодії алгоритму Дейкстри, алгоритму A та алгоритму пошуку точок переходу (Jump Point Search, JPS) на двовимірних сіткових структурах. Для формування тестових середовищ реалізовано генерацію лабіринтів на основі алгоритму бінарного дерева та рандомізованого алгоритму Прима, а також побудову мап із різною щільністю перешкод шляхом модифікації згенерованих структур. Порівняння алгоритмів здійснювалось за сумарним часом виконання в ході серії з 1000 ітерацій пошуку між випадково обраними парами точок на сітках розміром 255×255 клітинок. Визначено залежність ефективності алгоритмів від топології простору та щільності перешкод, сформульовано рекомендації щодо вибору алгоритму залежно від характеристик середовища. Отримані результати можуть бути використані під час проєктування навігаційних систем, ігрових застосунків та робототехнічних комплексів.*

Ключові слова: *пошук маршруту; алгоритм Дейкстри; алгоритм A*; Jump Point Search; евристичний пошук; двовимірна сітка; оптимізація маршрутів; генерація лабіринтів; навігаційні системи.*

Вступ

У сучасних інформаційних системах задачі пошуку оптимальних маршрутів посідають важливе місце та знаходять широке застосування у різних прикладних сферах, зокрема в навігаційних системах, робототехніці, телекомунікаційних мережах і індустрії відеоігор. Ефективність розв’язання таких задач безпосередньо впливає на продуктивність програмного забезпечення, особливо в умовах обмежених обчислювальних ресурсів та необхідності обробки даних у реальному часі.

Теоретичною основою задач маршрутизації є теорія графів, у межах якої простір моделюється у вигляді множини вершин та зв’язків між ними. Такий підхід дозволяє формалізувати процес пошуку шляху та застосовувати ефективні алгоритмічні методи для знаходження оптимальних або близьких до оптимальних рішень.

Особливий інтерес становить застосування алгоритмів пошуку маршруту у відеоіграх, де вони використовуються для керування рухом персонажів, побудови траєкторій та моделювання поведінки об’єктів. На відміну від класичних задач маршрутизації, у цьому середовищі додатково накладаються обмеження, пов’язані з необхідністю швидкого прийняття рішень, динамічністю середовища та великою

кількістю одночасних запитів на пошук шляхів. У більшості випадків ігровий простір моделюється у вигляді двовимірної прямокутної сітки, що визначає специфіку застосовуваних алгоритмів.

Актуальність дослідження зумовлена необхідністю підвищення ефективності алгоритмів пошуку маршруту в умовах обмежених ресурсів і різноманітних структур середовища. Незважаючи на значну кількість відомих алгоритмів, їх практична ефективність суттєво залежить від характеристик простору пошуку, таких як структура перешкод, щільність середовища та топологія графа.

Метою даного дослідження є порівняльний аналіз швидкодії поширених алгоритмів пошуку маршруту та визначення їх ефективності залежно від типу середовища на двовимірній сітці.

Для досягнення поставленої мети сформульовано такі завдання:

– реалізувати алгоритми генерації лабіринтів різних типів;
– реалізувати методи побудови мап на основі згенерованих лабіринтів;
– обґрунтувати вибір технологій для представлення та обробки середовища;
– реалізувати алгоритми пошуку маршруту, зокрема алгоритм Дейкстри, A* та алгоритм пошуку точок переходу;
– розробити методику експериментального порівняння алгоритмів;
– провести експериментальні дослідження на різних типах середовищ;
– здійснити аналіз отриманих результатів і визначити області ефективного застосування кожного алгоритму.

Наукова новизна роботи полягає у проведенні системного порівняльного аналізу алгоритмів пошуку маршруту в умовах різних типів структурованих середовищ (лабіринтів і мап), з урахуванням впливу топології простору на їх швидкодію.

Практичне значення отриманих результатів полягає у можливості використання сформульованих рекомендацій для вибору ефективних алгоритмів пошуку маршруту в задачах розробки програмного забезпечення, зокрема в ігрових застосуваннях та системах, що працюють у реальному часі.

Виклад основного матеріалу

Для реалізації поставлених у роботі задач було визначено низку вимог до програмних засобів та технологій, зокрема: підтримка складних структур даних, можливість ефективної роботи з графовими моделями та двовимірними сітками, наявність засобів для обробки растрових зображень, кросплатформність, а також достатня гнучкість для реалізації алгоритмів пошуку маршруту.

З урахуванням зазначених вимог як основний інструмент розробки було обрано мову програмування Python [3]. Вибір Python зумовлений її широким використанням у сфері розробки алгоритмічних та дослідницьких програмних рішень, наявністю значної кількості спеціалізованих бібліотек, а також підтримкою об’єктно-орієнтованого та функціонального підходів до програмування. Крім того, Python забезпечує зручні засоби роботи зі структурами даних, що є важливим під час реалізації алгоритмів пошуку на графах і сіткових середовищах.

Для роботи із зображеннями використано бібліотеку Pillow, яка є розширенням бібліотеки Python Imaging Library (PIL). Дана бібліотека забезпечує можливість завантаження, створення, редагування та збереження растрових зображень у різних графічних форматах. Використання Pillow дозволило представляти лабіринти та мапи у вигляді двовимірних масивів пікселів, що спрощує реалізацію алгоритмів генерації середовища та візуалізації знайдених маршрутів.

Для збереження сформованих зображень було використано формат BMP із кольоровою моделлю RGB. Такий підхід забезпечує коректне та наочне відображення

побудованих маршрутів на фоні лабіринтів і мап, а також мінімізує втрати якості під час збереження графічних даних.

1. Алгоритми побудови

1.1. Побудова лабіринту на основі бінарного дерева

Одним із базових методів генерації лабіринтів є алгоритм побудови на основі бінарного дерева. Даний підхід характеризується простотою реалізації та забезпечує формування зв'язного лабіринту без циклів.

На початковому етапі формується прямокутна сітка, у якій прохідні клітинки та стіни розташовуються почергово. Прохідні області позначаються світлими клітинками, тоді як непрохідні ділянки – темними.

Подальша генерація лабіринту виконується шляхом послідовного обходу всіх прохідних клітинок. Для кожної клітинки випадковим чином обирається один із двох можливих напрямків – угору або праворуч. Якщо клітинка розташована на межі сітки та один із напрямків недоступний, вибір здійснюється лише серед допустимих варіантів. Після визначення напрямку відповідна стіна видаляється, що створює новий прохід між сусідніми клітинками.

У результаті виконання алгоритму формується однозв'язний лабіринт без замкнених циклів, у якому між будь-якими двома точками існує єдиний оптимальний маршрут. Характерною особливістю такого методу є наявність структурного зміщення у напрямку верхнього правого кута, що проявляється у переважній орієнтації проходів та відкритості верхньої і правої меж лабіринту (рис. 1).

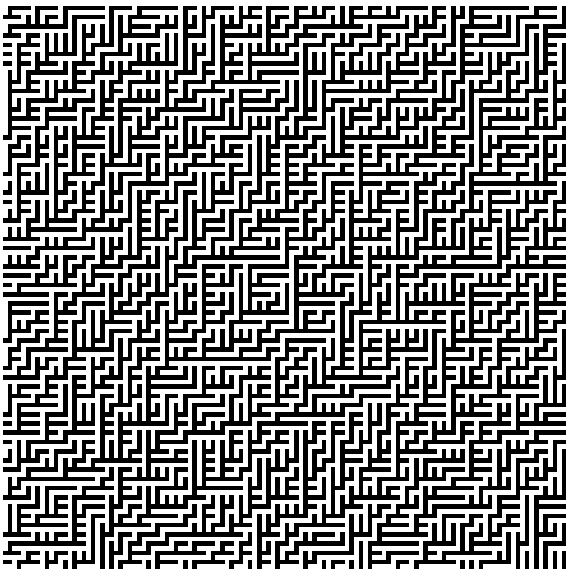


Рис. 1. Лабіринт, побудований на основі алгоритму бінарного дерева

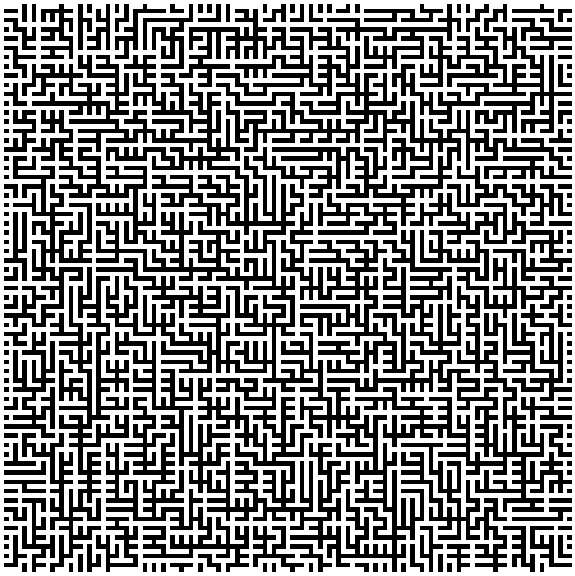


Рис. 2. Мапа на основі лабіринту з рис. 1

1.2. Побудова лабіринту на основі рандомізованого алгоритму Прима

Алгоритм Прима належить до жадібних алгоритмів та використовується для побудови мінімального кістякового дерева у зв'язному неорієнтованому графі. Для генерації лабіринтів застосовується його рандомізована модифікація, яка дозволяє формувати випадкові, але зв'язні структури проходів.

Побудова лабіринту починається із сітки, повністю заповненої стінами. На першому етапі випадковим чином обирається стартова клітинка, яка додається до

множини прохідних точок лабіринту. Після цього всі суміжні зі стартовою клітинкою стіни заносяться до окремого списку.

Далі алгоритм працює ітеративно. Із переліку стін випадковим чином вибирається одна стіна, після чого перевіряється можливість створення проходу через неї до нової клітинки. Якщо така клітинка ще не належить лабіринту, стіна видаляється, а нова клітинка додається до множини прохідних точок. Після цього до списку додаються всі прилеглі до нової клітинки стіни. Процес повторюється доти, доки список доступних стін не стане порожнім.

У результаті формується лабіринт із більш природною та менш регулярною структурою порівняно з лабіринтом, побудованим на основі бінарного дерева. Як показано на рис. 2, для такого лабіринту не характерне виражене зміщення у певному напрямку, а також відсутні порожні області вздовж меж сітки. Крім того, у структурі можуть виникати хрестоподібні перехрестя та складніші конфігурації проходів, що не зустрічаються при використанні бінарного алгоритму.

Важливою особливістю алгоритму є те, що сформований лабіринт залишається однозв'язним, тобто між будь-якими двома точками існує шлях. При цьому структура не містить ізольованих областей та циклів, що відповідає властивостям мінімального кістякового дерева.

Використання двох різних підходів до генерації лабіринтів – алгоритму бінарного дерева та рандомізованого алгоритму Прима – дозволяє отримати середовища з різною структурою проходів і характером перешкод. Це створює достатню основу для подальшого порівняльного аналізу алгоритмів пошуку маршруту, а також для побудови більш складних карт, що розглядаються в наступному підрозділі.

1.3. Побудова мап на основі лабіринтів

Лабіринти є зручним середовищем для тестування алгоритмів пошуку маршруту, однак вони не повністю відображають структуру реальних ігрових або навігаційних карт. У практичних задачах маршрутизації часто зустрічаються як вузькі проходи, так і відкриті області з перешкодами різної форми та густоти. Тому для формування більш універсальних карт у даній роботі використано підхід, заснований на модифікації згенерованих лабіринтів.

Метод побудови карт складається з декількох послідовних етапів та базується на поєднанні процедур генерації лабіринтів і подальшої трансформації їх структури.

На першому етапі формується базовий лабіринт. Для прикладу використовується лабіринт, побудований на основі бінарного дерева (див. рис. 1).

Другий етап передбачає видалення частини тупикових проходів з метою зменшення густоти лабіринту. Для цього виконується послідовний перегляд усіх прохідних клітинок. Якщо клітинка має лише одного сусіда, вона вважається тупиковою та додається до списку на видалення. Після завершення обходу всі знайдені тупикові точки видаляються. Процедура може повторюватися декілька разів залежно від бажаної структури карти. У наведеному прикладі операція виконувалась тричі.

Кількість ітерацій безпосередньо впливає на форму та ширину проходів у майбутній карті. Крім того, на рис. 3 можна помітити характерне зміщення структури у напрямку верхнього правого кута, властиве лабіринтам, побудованим за бінарним алгоритмом.

Наступним етапом є розширення проходів методом «нарощення». Для цього аналізуються всі клітинки поточного лабіринту та формується список прилеглих стін. Якщо стіна має три або більше сусідніх прохідних клітинок, вона перетворюється на прохід і додається до структури карти. Такий підхід дозволяє збільшити ширину коридорів та створити відкриті області.

Кількість повторень цього процесу визначає ступінь відкритості карти та густоту перешкод. Після завершення процедури, за необхідності, може повторно виконуватись видалення тупикових ділянок для згладжування форми проходів та усунення надлишкових вузьких областей.

На рис. 3 показано бінарний лабіринт (з рис. 1) після видалення кількох тупиків, що дозволяє зробити лабіринт менш щільним.

На рис. 4 представлена карта, отримана після нарощення проходів на основі бінарного лабіринту з рис. 1. Видно, що напрямок розширення коридорів виражається у продовгуватих перешкодах.

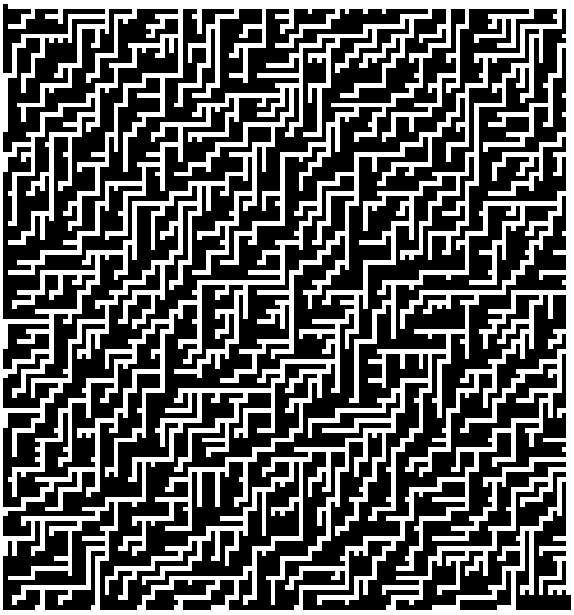


Рис. 3. Лабіринт з рис. 1 після трьох ітерацій видалення глухих кутів.

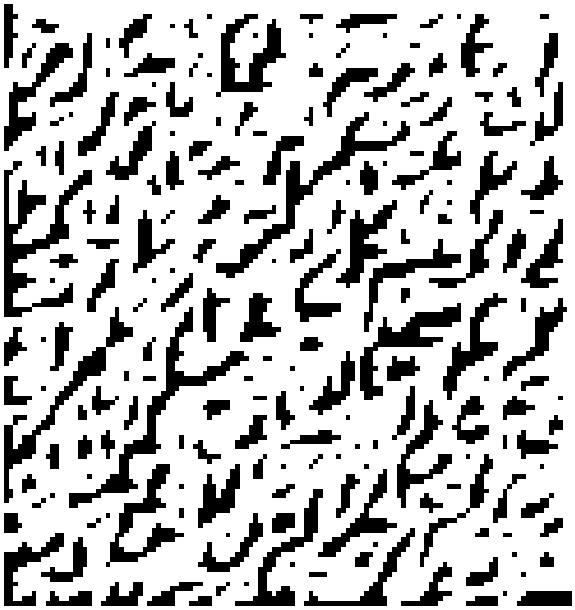


Рис. 4. Мапа на основі лабіринту з рис. 1

На рис. 5 представлено результат модифікації лабіринту, побудованого на основі рандомізованого алгоритму Прима (див. рис. 2), після кількох ітерацій видалення тупикових проходів. Застосування цієї процедури дозволяє суттєво зменшити густоту вузьких коридорів та сформувати більш відкриту структуру середовища, придатну для подальшого аналізу алгоритмів маршрутизації.

На рис. 6 наведено карту, сформовану на основі лабіринту Прима після виконання етапу розширення проходів методом «нарощення». У результаті такої трансформації структура карти набуває більш природного вигляду, характерного для реальних ігрових або навігаційних середовищ, де присутні як вузькі проходи, так і відкриті області різної конфігурації.

На рис. 4 помітно, що особливості вихідного бінарного лабіринту впливають на форму перешкод, які мають виражену орієнтацію в одному напрямку.

Аналогічні процедури були виконані для лабіринту, побудованого за рандомізованим алгоритмом Прима.

Порівняння рис. 4 та рис. 6 демонструє, що структура кінцевої карти значною мірою залежить від алгоритму генерації початкового лабіринту. Карти, побудовані на основі алгоритму Прима, мають більш рівномірний розподіл перешкод та менш виражену орієнтацію проходів, тоді як карти на основі бінарного дерева характеризуються певною напрямленістю структури. Це дозволяє використовувати

сформовані карти для дослідження ефективності алгоритмів пошуку маршруту в середовищах різної складності та конфігурації.

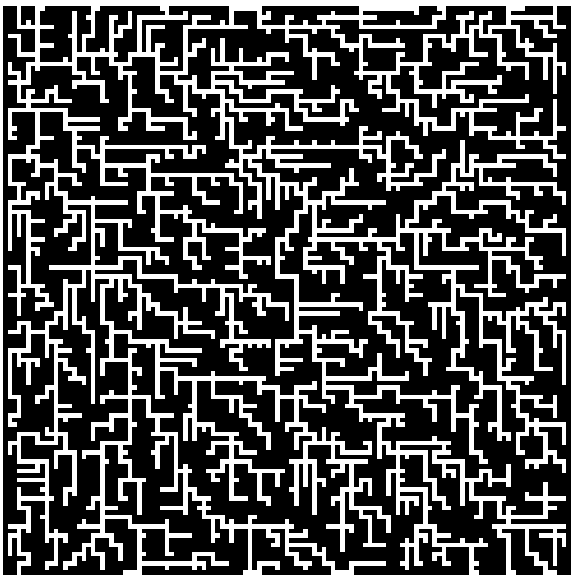


Рис. 5. Лабіринт на основі алгоритму Прима після трьох ітерацій видалення тупикових проходів.

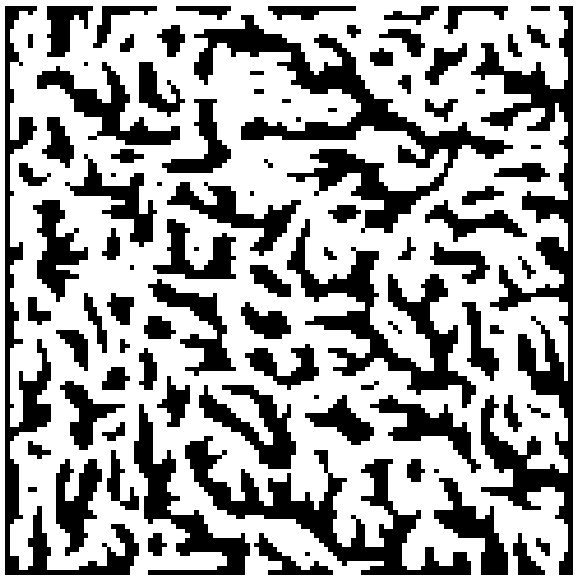


Рис. 6. Мапа на основі лабіринту з рис. 2

Порівняння карт, наведених на рис. 4 та рис. 6, демонструє суттєвий вплив початкового алгоритму генерації лабіринту на структуру кінцевого середовища. Карти, побудовані на основі бінарного дерева, характеризуються помітною напрямленістю та витягнутими перешкодами, тоді як карти, сформовані на основі алгоритму Прима, мають більш хаотичний та рівномірний розподіл прохідних зон і перешкод. Такі відмінності дозволяють формувати тестові середовища з різним рівнем складності та використовувати їх для комплексного дослідження ефективності алгоритмів пошуку маршруту.

2. Алгоритми пошуку маршруту

2.1. Алгоритм Дейкстри

Алгоритм Дейкстри є класичним алгоритмом теорії графів, призначеним для знаходження найкоротших шляхів від заданої початкової вершини до всіх інших вершин зваженого графа з невід'ємними вагами ребер [3].

Нехай граф задано як $G=(V,E)$, де V – множина вершин, а E – множина ребер. Алгоритм ґрунтується на ітеративному уточненні оцінок відстаней $d(v)$ від початкової вершини s до всіх інших вершин шляхом послідовної обробки вершин у порядку зростання поточної оцінки відстані.

У рамках даної роботи граф представляється у вигляді прямокутної сітки, де вершини відповідають клітинкам, а ребра – можливим переходам між ними. Вартість переходу між сусідніми клітинками задається як стала величина (для ортогональних і діагональних переміщень).

Узагальнений алгоритм Дейкстри можна подати у вигляді наступної послідовності кроків:

1. Ініціалізація: для всіх вершин $v \in V$ задається $d(v)=\infty$, для початкової вершини s – $d(s)=0$.
2. Початкова вершина додається до черги з пріоритетом, де пріоритет визначається значенням $d(v)$.
3. Поки черга не порожня:
   - обирається вершина з мінімальним значенням $d(v)$;
   - для кожної суміжної вершини виконується операція релаксації: $d(u)=min(d(u),d(v)+w(v,u))$, де $w(v,u)$ – вага ребра;
   - у разі оновлення значення $d(u)$ відповідна вершина додається до черги.

Після завершення обчислень маршрут до цільової вершини відновлюється за допомогою збережених попередників.

У даній реалізації алгоритм виконується без дострокового завершення після досягнення цільової вершини, що дозволяє отримати повну інформацію про найкоротші шляхи до всіх доступних вершин графа.

Основною перевагою алгоритму є його універсальність і гарантована оптимальність знайдених шляхів. Водночас його суттєвим недоліком є висока обчислювальна складність у задачах з великим простором пошуку, оскільки алгоритм не використовує інформацію про розташування цільової вершини.

На рис. 7 наведено приклад маршруту, знайденого алгоритмом Дейкстри.

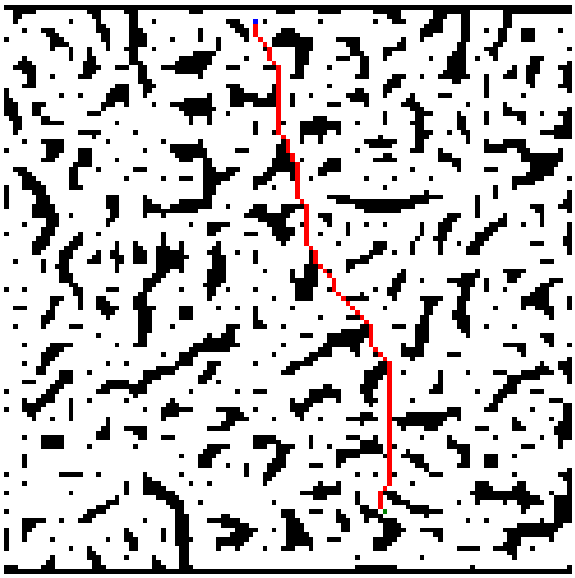


Рис. 7. Маршрут, побудований алгоритмом Дейкстри

2.2. Алгоритм A*

Алгоритм A* належить до класу евристичних алгоритмів пошуку та використовується для знаходження найкоротшого шляху між двома вершинами графа з невід'ємними вагами ребер [4]. Алгоритм було запропоновано у 1968 році та він поєднує властивості алгоритму Дейкстри та жадібного пошуку.

Основною особливістю алгоритму A* є використання оціночної функції:

$$f(n) = g(n) + h(n),$$

де $g(n)$ – вартість шляху від початкової вершини до поточної вершини n, а $h(n)$ – евристична оцінка вартості шляху від вершини n до цільової.

За умови допустимості евристичної функції (тобто $h(n)$ не переоцінює реальну

відстань), алгоритм гарантує знаходження оптимального маршруту.

Нами для евристичної оцінки використано евклідову відстань між вершинами, що є доцільним у випадку, коли дозволені діагональні переміщення на сітці.

Алгоритм A* функціонує аналогічно до алгоритму Дейкстри, однак вибір наступної вершини для обробки здійснюється з урахуванням значення функції $f(n)$, що дозволяє спрямовувати пошук у напрямку цільової вершини та зменшувати кількість розглянутих вершин.

На відміну від алгоритму Дейкстри, у даній реалізації A* пошук припиняється одразу після досягнення цільової вершини, що дозволяє додатково скоротити час виконання.

Основною перевагою алгоритму є значно вища ефективність порівняно з алгоритмом Дейкстри за рахунок використання евристики. Крім того, алгоритм є гнучким у налаштуванні, оскільки вибір евристичної функції дозволяє адаптувати поведінку пошуку до конкретних умов задачі.

На рис. 8 наведено приклад маршруту, знайденого алгоритмом A*.

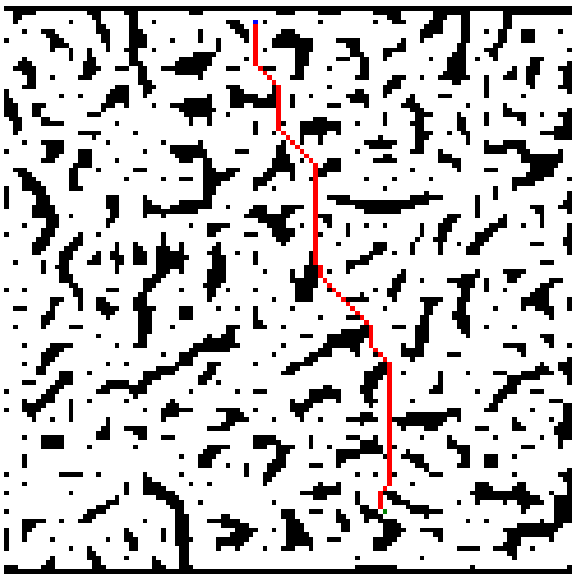


Рис. 8. Маршрут, побудований алгоритмом A*

Порівняння маршрутів, зображених на рис. 7 та рис. 8, показує, що, незважаючи на однакову довжину, отримані шляхи можуть відрізнятися за конфігурацією, що зумовлено різною стратегією обходу простору пошуку.

2.3. Алгоритм пошуку точок переходу

Алгоритм пошуку точок переходу (англ. Jump Point Search, JPS) є оптимізацією алгоритму A*, орієнтованою на роботу з прямокутними сітками з рівномірною вартістю переміщення. Запропонований у 2011 році, цей підхід спрямований на зменшення кількості розглянутих вершин шляхом усунення симетричних варіантів маршрутів та обмеження області пошуку лише ключовими точками.

Основна ідея алгоритму полягає у скороченні простору пошуку за рахунок розгляду не всіх сусідніх вершин, як у класичних алгоритмах, а лише так званих точок переходу (jump points), які визначаються відповідно до певних правил поширення. Це дозволяє уникнути надлишкових обчислень і значно підвищити ефективність пошуку.

Алгоритм базується на двох ключових механізмах: відсічення сусідів та виявлення вимушених сусідів.

Відсікання сусідів полягає у виключенні з розгляду тих сусідніх вершин, до яких

існує альтернативний шлях такої ж або меншої довжини через інші вершини. Таким чином, алгоритм уникає дослідження симетричних маршрутів, що не впливають на оптимальність результату.

Вимушені сусіди виникають у випадках, коли наявність перешкод змінює структуру допустимих шляхів. Якщо оптимальний маршрут до певної вершини може проходити лише через поточну вершину, така вершина повинна бути включена до розгляду незалежно від загальних правил відсічення.

Процес дослідження простору пошуку в алгоритмі JPS має спрямований характер. Як показано на рис. 9, розширення виконується переважно у напрямках, визначених попереднім кроком, причому переміщення по горизонталі та вертикалі мають пріоритет над діагональними. Діагональний рух здійснюється лише після вичерпання можливостей у відповідних ортогональних напрямках.

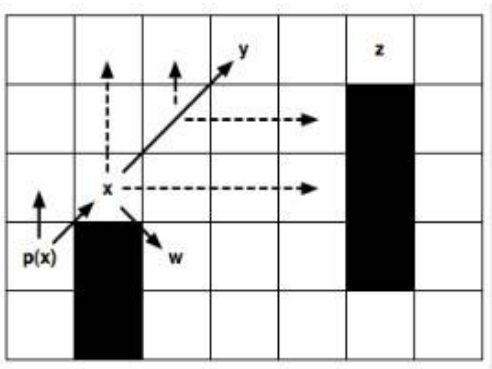


Рис. 9. Приклад дослідження точок

Точка переходу визначається як вершина, що:

– має хоча б одного вимушеного сусіда;
– є результатом діагонального переміщення, після якого можливе подальше розгалуження пошуку;
– є початковою або цільовою вершиною.

На відміну від алгоритму A*, у якому до черги з пріоритетом додаються всі потенційні вершини, в алгоритмі JPS до розгляду включаються лише точки переходу. Пріоритет визначається аналогічно до A*, як:

$$f(n) = g(n) + h(n).$$

Узагальнений алгоритм можна подати таким чином:

1. Ініціалізація: усім прохідним вершинам призначається початкове значення відстані (нескінченність), початковій вершині – нуль.
2. Початкова вершина додається до черги з пріоритетом.
3. Поки черга не порожня:
   – обирається вершина з мінімальним значенням f(n);
   – виконується пошук точок переходу відповідно до правил поширення;
   – знайдені точки переходу додаються до черги.

Після досягнення цільової вершини здійснюється відновлення маршруту.

Завдяки описаним механізмам алгоритм JPS забезпечує суттєве скорочення кількості досліджуваних вершин порівняно з алгоритмами Дейкстри та A*, що позитивно впливає на швидкодію. Водночас слід зазначити, що його застосування

обмежується середовищами, які можуть бути представлені у вигляді регулярної сітки з однаковою вартістю переміщення.

На рис. 10 наведено приклад маршруту, побудованого за допомогою алгоритму пошуку точок переходу. Як видно, структура маршруту може відрізнятися від результатів інших алгоритмів, однак довжина шляху залишається оптимальною. Зокрема, у випадку симетричних альтернатив алгоритм надає перевагу діагональним переміщенням, що впливає на геометрію отриманого маршруту.

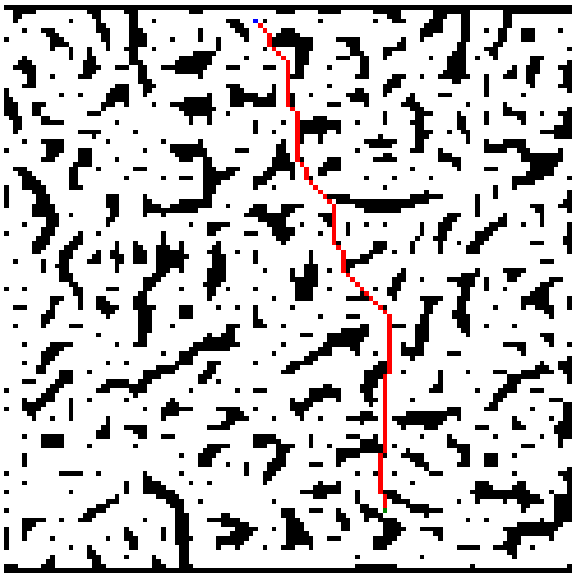


Рис. 10. Маршрут, знайдений алгоритмом пошуку точок переходу

3. Порівняння алгоритмів

Для оцінювання було використано два типи лабіринтів розміром 255×255 клітинок: лабіринт, згенерований на основі бінарного дерева, та лабіринт, побудований за допомогою рандомізованого алгоритму Прима. Основним критерієм порівняння обрано час виконання алгоритмів при пошуку маршруту між випадково обраними парами точок.

3.1. Методика порівняння на лабіринтах

Експериментальне дослідження проводилося за уніфікованою процедурою, що забезпечує коректність порівняння алгоритмів. Для кожного типу лабіринту виконувалася наступна послідовність дій:

1. Генерація лабіринту відповідного типу.
2. Ініціалізація змінних для накопичення часу виконання кожного алгоритму.
3. Встановлення кількості ітерацій експерименту (1000).
4. Для кожної ітерації:
   - випадковим чином обиралися стартова та цільова вершини;
   - виконувалося знаходження маршруту за допомогою алгоритмів Дейкстри, A* та пошуку точок переходу (ПТП);
   - фіксувався час виконання кожного алгоритму з подальшим накопиченням.

Після завершення серії ітерацій обчислювався сумарний час роботи кожного алгоритму.

Такий підхід дозволяє мінімізувати вплив випадкових факторів і отримати узагальнену оцінку продуктивності.

3.2. Лабіринт на основі бінарного дерева

Перший експеримент було проведено на лабіринті, сформованому за алгоритмом бінарного дерева (рис. 11), який характеризується переважно лінійною структурою проходів та наявністю вираженого напрямку.

За результатами експерименту отримано такі значення сумарного часу виконання:

- алгоритм Дейкстри – 0:20:25.041311;
- алгоритм A* – 0:11:16.526667;
- алгоритм ПТП – 0:07:06.049409.

Отримані результати демонструють, що алгоритм Дейкстри має найнижчу продуктивність, що зумовлено повним обходом доступної області графа без урахування цільової вершини. Використання евристичної функції в алгоритмі A* дозволяє суттєво обмежити область пошуку, що забезпечує приблизно дворазове скорочення часу виконання.

Найкращий результат продемонстрував алгоритм пошуку точок переходу. Це пояснюється тим, що в умовах переважно прямолінійних проходів він ефективно усуває симетричні варіанти маршрутів і зменшує кількість розглянутих вершин.

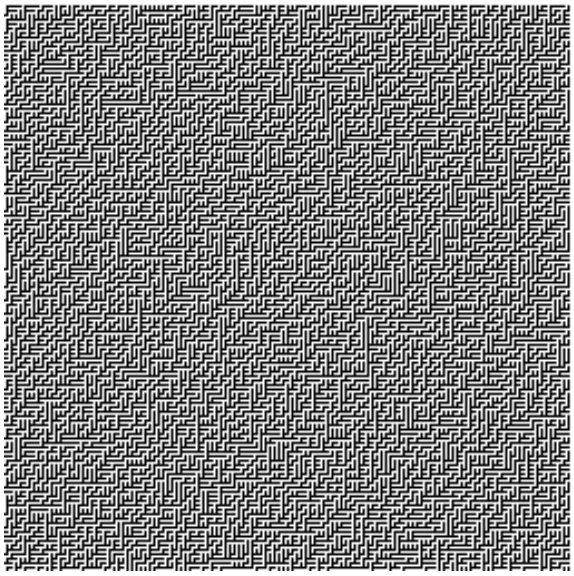


Рис. 11. Лабіринт на основі бінарного дерева

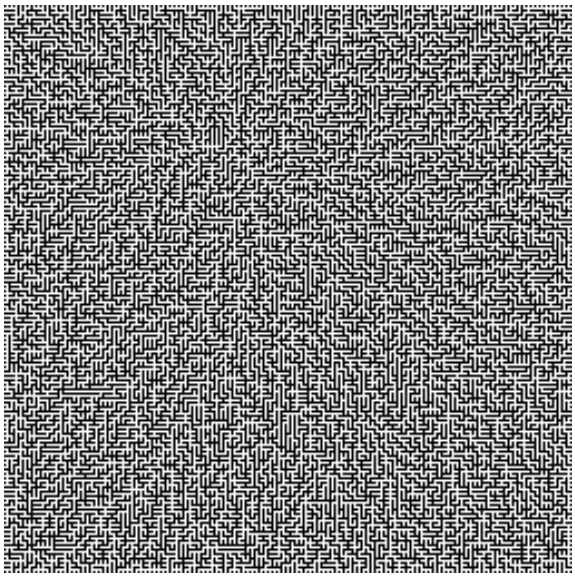


Рис. 12. Лабіринт на основі алгоритму Прима

3.3. Лабіринт на основі алгоритму Прима

Другий експеримент було проведено на лабіринті, згенерованому за допомогою алгоритму Прима (рис. 12), який відзначається більш складною топологією, наявністю численних перехресть і відсутністю домінуючого напрямку.

Результати вимірювань мають наступний вигляд:

- алгоритм Дейкстри – 0:20:13.978304;
- алгоритм A* – 0:11:02.435265;
- алгоритм ПТП – 0:04:35.152273.

Аналогічно до попереднього випадку, алгоритм Дейкстри продемонстрував найменшу ефективність, що підтверджує його обмеженість у задачах з великим простором пошуку. Алгоритм A* знову показав значно кращі результати завдяки використанню евристичної оцінки.

Водночас у даному типі лабіринту спостерігається ще більш виражена перевага алгоритму ПТП. Це можна пояснити тим, що складна структура середовища з великою кількістю перешкод створює умови, у яких механізми відсічення та виявлення вимушених сусідів працюють найбільш ефективно, суттєво скорочуючи кількість досліджуваних вершин.

Результати експерименту наведено в табл. 1.

Таблиця 1

Порівняння часу виконання алгоритмів на лабіринтах

Тип лабіринту	**Алгоритм Дейкстри**	**A***	**ПТП**
Бінарне дерево	0:20:25	0:11:16	0:07:06
Алгоритм Прима	0:20:13	0:11:02	0:04:35

Порівняльний аналіз показує, що продуктивність алгоритмів пошуку маршруту істотно залежить від структури середовища. Алгоритм Дейкстри демонструє стабільно низьку швидкодію через відсутність спрямованості пошуку. Алгоритм A* забезпечує значне покращення завдяки використанню евристики, проте його ефективність обмежується необхідністю розгляду значної кількості вершин у складних середовищах.

Алгоритм пошуку точок переходу виявився найбільш ефективним у всіх досліджених випадках. Його перевага зумовлена зменшенням симетрії пошуку та концентрацією обчислень лише на ключових вершинах. Отримані результати підтверджують доцільність використання цього алгоритму для задач пошуку маршруту на регулярних сітках з рівномірною вартістю переміщення.

4. Порівняння на мапах

Наступним етапом експериментального дослідження стало порівняння ефективності алгоритмів пошуку маршруту в середовищах, що моделюють більш реалістичні умови порівняно з лабіринтами. Для цього було використано чотири мапи з різною щільністю перешкод, побудовані на основі двох типів структур: лабіринту Прима та бінарного лабіринту. Для кожного типу розглядалися два варіанти: з меншою та більшою щільністю перешкод.

Метою дослідження є оцінка впливу структури середовища та ступеня його заповненості на продуктивність алгоритмів Дейкстри, A* та пошуку точок переходу (ПТП).

4.1. Методика експерименту

Методика проведення експерименту є аналогічною до описаної в підрозділі 3.1. Для кожної мапи виконувалася серія з 1000 ітерацій, у межах яких:

1) випадковим чином обиралися стартова та цільова вершини;
2) кожен алгоритм виконував пошук маршруту;
3) фіксувався час виконання з подальшим накопиченням.

Після завершення серії ітерацій обчислювався сумарний час роботи кожного алгоритму, що використовувався як інтегральна характеристика продуктивності.

4.2. Отримані результати

Результати експерименту для ***мапи з відносно низькою щільністю перешкод***, побудованої на основі алгоритму Прима (рис. 13), наведено нижче:

- алгоритм Дейкстри – 0:33:21.470127;
- алгоритм A* – 0:18:31.745551;
- алгоритм ПТП – 0:03:58.323529.

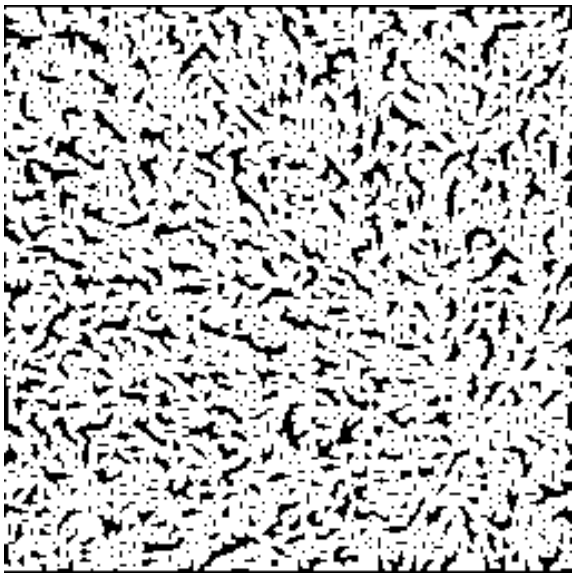


Рис. 13. Мапа на основі лабіринту Прима з меншою щільністю перешкод

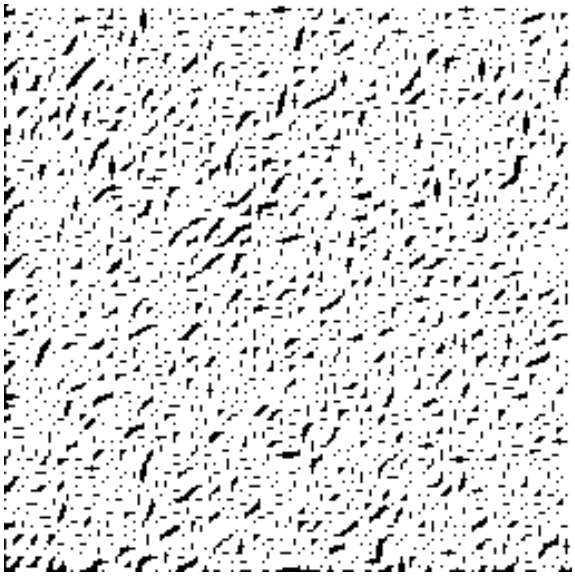


Рис. 14. Мапа на основі бінарного лабіринту з меншою щільністю перешкод

Отримані результати свідчать про суттєву перевагу алгоритму ПТП у середовищах із великою кількістю відкритих ділянок. У таких умовах алгоритм ефективно скорочує простір пошуку за рахунок руху між ключовими точками, ігноруючи проміжні вершини. Алгоритм Дейкстри, навпаки, демонструє найгірші результати через необхідність повного обходу доступної області.

Для ***мапи, сформованої на основі бінарного лабіринту з меншою щільністю перешкод*** (рис. 14), отримано такі результати:

- алгоритм Дейкстри – 0:37:19.253132;
- алгоритм A* – 0:21:08.208494;
- алгоритм ПТП – 0:04:17.588614.

Аналогічно до попереднього випадку, алгоритм ПТП демонструє найвищу ефективність. При цьому варто зазначити, що структура мапи, успадкована від бінарного лабіринту, зумовлює часткову впорядкованість перешкод, що дещо знижує ефективність евристичного спрямування алгоритму A* порівняно з більш хаотичними структурами.

Для ***мапи з підвищеною щільністю перешкод, побудованої на основі алгоритму Прима*** (рис. 15), отримано такі значення:

- алгоритм Дейкстри – 0:26:59.961727;
- алгоритм A* – 0:15:29.628627;
- алгоритм ПТП – 0:04:06.620957.

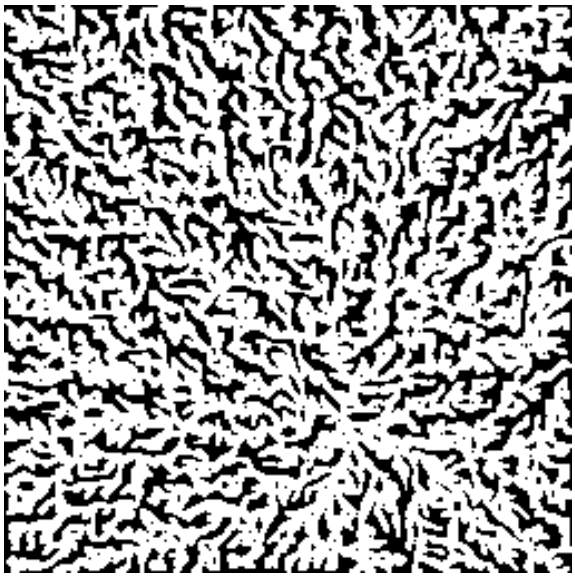


Рис. 15. Мапа на основі лабіринту Прима з більшою щільністю перешкод

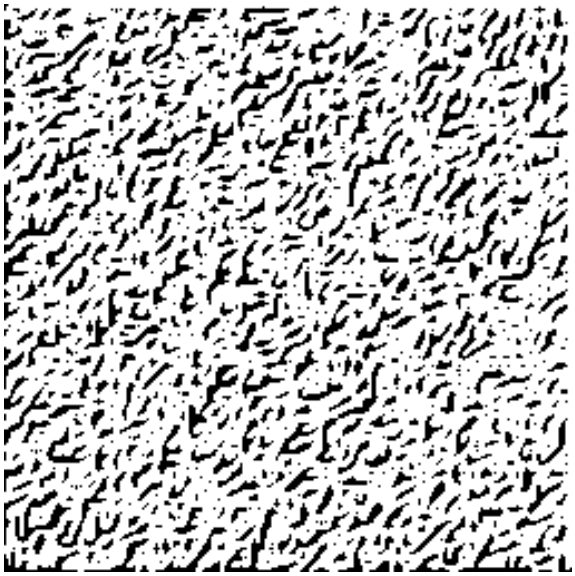


Рис. 16. Мапа на основі бінарного лабіринту з більшою щільністю перешкод

Збільшення щільності перешкод призводить до зменшення області доступного простору, що частково знижує обчислювальне навантаження для всіх алгоритмів. У цих умовах алгоритм A* демонструє помітне покращення ефективності порівняно з алгоритмом Дейкстри, оскільки евристична функція дозволяє більш точно спрямовувати пошук. Алгоритм ПТП зберігає найкращі показники завдяки ефективному скороченню кількості розглянутих вершин.

Останній експеримент проведено на ***мапі, побудованій на основі бінарного лабіринту з високою щільністю перешкод*** (рис. 16):

- алгоритм Дейкстри – 0:31:03.161842;
- алгоритм A* – 0:17:09.005322;
- алгоритм ПТП – 0:04:12.425184.

У цьому випадку спостерігається аналогічна тенденція: алгоритм ПТП демонструє найкращу продуктивність, тоді як алгоритм Дейкстри залишається найменш ефективним. Висока щільність перешкод обмежує варіативність маршрутів, що частково підвищує ефективність алгоритму A*, однак не дозволяє йому досягти рівня ПТП.

Аналіз результатів експерименту дозволяє зробити такі ***узагальнення***:

1) продуктивність алгоритмів суттєво залежить від структури середовища та щільності перешкод;
2) алгоритм Дейкстри демонструє найнижчу ефективність у всіх досліджених випадках через відсутність спрямованості пошуку;
3) алгоритм A* забезпечує стабільне покращення продуктивності завдяки використанню евристичної функції, причому його ефективність зростає зі збільшенням щільності перешкод;
4) алгоритм пошуку точок переходу є найбільш ефективним у всіх експериментальних сценаріях, особливо у середовищах із великими відкритими просторами.

Отримані результати підтверджують доцільність використання алгоритму ПТП для задач пошуку маршруту на регулярних сітках, а також демонструють важливість

врахування характеристик середовища при виборі алгоритму.

Таблиця 2

Порівняння сумарного часу виконання алгоритмів та відносного прискорення

Тип середовища	Дейкстра	A*	ПТП	Прискорення A* відносно Дейкстри	Прискорення ПТП відносно A*	Прискорення ПТП відносно Дейкстри
Бінарний лабіринт	20:25	11:16	7:06	44,8%	36,9%	65,2%
Лабіринт Прима	20:13	11:02	4:35	45,4%	58,5%	77,3%
Мапа (Прима, мало перешкод)	33:21	18:31	3:58	44,5%	78,6%	88,1%
Мапа (бінарна, мало перешкод)	37:19	21:08	4:17	43,4%	79,7%	88,5%
Мапа (Прима, багато перешкод)	26:59	15:29	4:06	42,6%	73,5%	84,8%
Мапа (бінарна, багато перешкод)	31:03	17:09	4:12	44,8%	75,5%	86,5%

Висновки

У роботі здійснено реалізацію та комплексне дослідження алгоритмів генерації лабіринтів і мап, а також алгоритмів пошуку маршрутів, зокрема алгоритму Дейкстри, алгоритму A* та алгоритму пошуку точок переходу (Jump Point Search, JPS). Проведене експериментальне дослідження дозволило оцінити їхню швидкодію в умовах різних типів середовищ, представлених у вигляді двовимірних сіток із різною структурою та щільністю перешкод.

На основі отриманих результатів можна сформулювати такі узагальнення:

1. Порівняльна ефективність алгоритмів.

Встановлено, що алгоритм A* демонструє стабільне покращення швидкодії порівняно з алгоритмом Дейкстри, забезпечуючи скорочення часу виконання в середньому на 42-47%. Це підтверджує доцільність використання евристичних підходів у задачах маршрутизації.

2. Переваги алгоритму пошуку точок переходу.

Алгоритм JPS показав найвищу ефективність серед розглянутих методів, забезпечуючи скорочення часу пошуку до 65–88% порівняно з алгоритмом Дейкстри та суттєво випереджаючи алгоритм A*. Найбільший виграш у швидкодії спостерігається у середовищах зі складною структурою та високою щільністю перешкод, що пояснюється зменшенням кількості симетричних обчислень.

3. Вплив структури середовища на продуктивність.

Експериментально підтверджено, що ефективність алгоритмів суттєво залежить від топології простору. Для середовищ із меншою щільністю перешкод різниця між алгоритмами зменшується, тоді як у більш складних середовищах переваги евристичних і оптимізованих підходів стають більш вираженими.

4. Рекомендації щодо застосування алгоритмів.

- алгоритм JPS доцільно використовувати у задачах із високими вимогами до швидкодії та частою зміною стартових і цільових позицій;
- алгоритм A* є ефективним компромісним рішенням у випадках, коли застосування JPS обмежене умовами задачі або структурою середовища;
- алгоритм Дейкстри доцільно застосовувати у задачах, що передбачають багаторазове використання результатів пошуку від однієї фіксованої вершини або за відсутності коректної евристичної функції.

5. Обмеження дослідження.

У роботі розглядалися виключно алгоритми, що забезпечують знаходження оптимального маршруту. Це зумовлює додаткові обчислювальні витрати, які можуть бути надмірними для задач, де допускається наближене розв'язання.

6. Практичне значення та перспективи подальших досліджень.

Отримані результати можуть бути використані при виборі алгоритмів пошуку маршруту в прикладних системах, зокрема у відеоіграх, робототехніці та навігаційних застосуваннях. Перспективним напрямом подальших досліджень є аналіз алгоритмів, що знаходять неоптимальні, але швидші маршрути, а також адаптація розглянутих методів до динамічних середовищ і тривимірних просторів.

Список використаної літератури

1. Severance, C. R. Python for Everybody: Exploring Data Using Python 3. CreateSpace Independent Publishing Platform, 2016. 244 p.
2. Miller, B. N., Ranum, D. L., and Anderson, J. Python Programming in Context. Jones & Bartlett Learning, 2019. 530 p.
3. Joshi, P. Artificial Intelligence with Python. Packt Publishing, 2017. 446 p.
4. Ni, Y.; Zhuo, Q.; Li, N.; Yu, K.; He, M.; Gao, X. Characteristics and Optimization Strategies of A* Algorithm and Ant Colony Optimization in Global Path Planning Algorithm. Int. J. Pattern Recognit. 2023, 37, 2351006.
5. Wang, P.; Liu, Y.; Yao, W.; Yu, Y. Improved A-star algorithm based on multivariate fusion heuristic function for autonomous driving path planning. Proc. Inst. Mech. Eng. Part D-J. Automob. Eng. 2022, 237, 1527-1542.
6. Zhang, Y.; Li, L.; Lin, H.; Ma, Z.; Zhao, J. Development of Path Planning Approach Using Improved A-star Algorithm in AGV System. J. Internet Technol. 2019, 20, 915-924.
7. Iram, N.; Amna, K.; Khurshid, A.; Zulfiqar, H. A Path-Planning Performance Comparison of RRT*-AB with MEA* in a 2-Dimensional Environment. Symmetry 2019, 11, 945.
8. Liu, L.; Lin, J.; Yao, J.; He, D.; Zheng, J.; Huang, J.; Shi, P. Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach. Wirel. Commun. Mob. Comput. 2021, 2021, 356-368.
9. Banerjee, N.; Chakraborty, S.; Raman, V.; Satti, S.R. Space Efficient Linear Time Algorithms for BFS, DFS and Applications. Theor. Comput. Syst. 2018, 62, 1736-1762.
10. Lai, W.K.; Shieh, C.S.; Yang, C.P. A D2D Group Communication Scheme Using Bidirectional and InCremental A-Star Search to Configure Paths. Mathematics 2022, 10, 3321.
11. Wang, H.; Qi, X.; Lou, S.; Jing, J.; He, H.; Liu, W. An Efficient and Robust Improved A* Algorithm for Path Planning. Symmetry 2021, 13, 2213.
12. Chen Yifu, Lu Wei, Ding Haojie. Research on Optimization Strategy of Dijkstra Algorithm // Computer Technology and Development. – 2006. – Vol. 16, No. 9. – pp. 73-75.
13. Zhang Yonglong. Optimization of Dijkstra Optimal Path Algorithm // Journal of Nanchang Institute of Technology. – 2006. – Vol. 25, No. 3. – pp. 30-33.
14. Using Prim's Algorithm to Generate Continuous Cave Maps [Електронний ресурс] // Режим доступу: https://kairumagames.com/blog/cavetutorial

References:

1. Severance, C. R. Python for Everybody: Exploring Data Using Python 3. CreateSpace Independent Publishing Platform, 2016. 244 p.
2. Miller, B. N., Ranum, D. L., and Anderson, J. Python Programming in Context. Jones & Bartlett Learning, 2019. 530 p.
3. Joshi, P. Artificial Intelligence with Python. Packt Publishing, 2017. 446 p.

4. Ni, Y.; Zhuo, Q.; Li, N.; Yu, K.; He, M.; Gao, X. Characteristics and Optimization Strategies of A* Algorithm and Ant Colony Optimization in Global Path Planning Algorithm. Int. J. Pattern Recognit. 2023, 37, 2351006.
5. Wang, P.; Liu, Y.; Yao, W.; Yu, Y. Improved A-star algorithm based on multivariate fusion heuristic function for autonomous driving path planning. Proc. Inst. Mech. Eng. Part D-J. Automob. Eng. 2022, 237, 1527-1542.
6. Zhang, Y.; Li, L.; Lin, H.; Ma, Z.; Zhao, J. Development of Path Planning Approach Using Improved A-star Algorithm in AGV System. J. Internet Technol. 2019, 20, 915–924.
7. Iram, N.; Amna, K.; Khurshid, A.; Zulfiqar, H. A Path-Planning Performance Comparison of RRT*-AB with MEA* in a 2-Dimensional Environment. Symmetry 2019, 11, 945.
8. Liu, L.; Lin, J.; Yao, J.; He, D.; Zheng, J.; Huang, J.; Shi, P. Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach. Wirel. Commun. Mob. Comput. 2021, 2021, 356–368.
9. Banerjee, N.; Chakraborty, S.; Raman, V.; Satti, S.R. Space Efficient Linear Time Algorithms for BFS, DFS and Applications. Theor. Comput. Syst. 2018, 62, 1736–1762.
10. Lai, W.K.; Shieh, C.S.; Yang, C.P. A D2D Group Communication Scheme Using Bidirectional and InCremental A-Star Search to Configure Paths. Mathematics 2022, 10, 3321.
11. Wang, H.; Qi, X.; Lou, S.; Jing, J.; He, H.; Liu, W. An Efficient and Robust Improved A* Algorithm for Path Planning. Symmetry 2021, 13, 2213.
12. Chen Yifu, Lu Wei, Ding Haojie. Research on Optimization Strategy of Dijkstra Algorithm // Computer Technology and Development. – 2006. – Vol. 16, No. 9. – pp. 73-75.
13. Zhang Yonglong. Optimization of Dijkstra Optimal Path Algorithm // Journal of Nanchang Institute of Technology. – 2006. – Vol. 25, No. 3. – pp. 30-33.
14. Using Prim's Algorithm to Generate Continuous Cave Maps // URL: https://kairumagames.com/blog/cavetutorial

HLADKA Liudmyla,
PhD in Physical and Mathematical Sciences, Associate Professor of the Department of Automation and Computer-Integrated Technologies, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
CHALYI Anton,
Software Developer, UniStar LLC, Smila, Ukraine

COMPARATIVE STUDY OF PATHFINDING ALGORITHM PERFORMANCE ON MAZE AND GRID-BASED ENVIRONMENTS

Summary. Introduction. *Pathfinding tasks on two-dimensional grid structures arise in a wide range of applied domains, including navigation systems, robotics, telecommunications, and video game development. The computational efficiency of pathfinding algorithms directly affects software performance, particularly under constrained resources and real-time processing requirements. Despite the availability of numerous well-known algorithms, their practical efficiency varies substantially depending on the structural properties of the search space, including obstacle topology, obstacle density, and graph connectivity. The selection of an appropriate algorithm for a given environment type remains an open practical problem.*

Purpose. *The purpose of this work is to conduct a comparative performance analysis of widely used pathfinding algorithms – Dijkstra's algorithm, the A* algorithm, and the Jump Point Search (JPS) algorithm – on two-dimensional grid environments of varying structure and obstacle density, and to formulate evidence-based recommendations for algorithm selection.*

Results. *The study was implemented in Python using the Pillow library for grid representation and BMP image generation. Two maze generation methods were employed: the Binary Tree algorithm, which produces mazes with predominantly linear corridor structures, and the randomized Prim's algorithm, which yields more complex, uniformly distributed topologies. Map environments of varying obstacle density were derived from both maze types through iterative dead-end removal and corridor expansion procedures, producing four distinct map configurations. Experimental evaluation was performed on 255×255 grids using 1000 randomized start–goal pairs per environment. Total accumulated execution time served as the primary performance metric. The results demonstrate that A* reduces execution time by 42-47% relative to Dijkstra's algorithm across all tested environments, owing to heuristic-guided search. The Jump Point Search algorithm achieved the highest performance in all scenarios, reducing total execution time by 65-88% relative to Dijkstra's algorithm and by 37-*

79% relative to A. The advantage of JPS was most pronounced on map environments with large open areas and low obstacle density, where symmetry pruning and jump point identification most effectively reduce the number of explored nodes. In high-density obstacle environments, the performance gain of JPS remained substantial, with A* also showing improved relative efficiency due to more focused heuristic guidance.*

Conclusion. *The experimental study confirms that heuristic and symmetry-based approaches substantially outperform classical graph traversal in grid pathfinding tasks. JPS is recommended for applications with high performance requirements and frequent re-planning between arbitrary node pairs. A* provides an effective compromise when JPS applicability is constrained by environmental conditions. Dijkstra's algorithm remains appropriate for scenarios requiring full shortest-path trees from a fixed source node or in the absence of a valid heuristic function. Future research directions include analysis of suboptimal but faster algorithms, extension to dynamic environments, and adaptation to three-dimensional search spaces.*

Keywords: *pathfinding algorithms; Dijkstra's algorithm; A* algorithm; Jump Point Search; heuristic search; two-dimensional grid; route optimization; maze generation; navigation systems; path planning.*

Одержано редакцією 07.06.2024 р.
Прийнято до публікації 28.08.2024 р.

УДК 004.932

DOI 10.31651/2076-5886-2024-1-21-33

PACS 07.05.Pj, 42.30.Va

ГАВРЮШЕНКО Анна Миколаївна
учитель математики та інформатики
Черкаської загальноосвітньої школи І-ІІІ ступенів № 32 Черкаської міської ради
e-mail: 91gavryushrenko@gmail.com
ORCID 0009-0000-0188-8505

БОРОЗДИХ Катерина Сергіївна
учениця Черкаської загальноосвітньої школи І-ІІІ ступенів № 32 Черкаської міської ради

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ВИЯВЛЕННЯ КЛЮЧОВИХ ТОЧОК НА ЗОБРАЖЕННЯХ

У статті розглянуто математичне підґрунтя методу SIFT як одного з перших та найбільш теоретично обґрунтованих підходів до розв'язання зазначеної задачі, а також проведено порівняльний аналіз шести сучасних методів: SIFT, SURF, BRISK, ORB, KAZE та AKAZE. Дослідження виконане на чотирьох синтетичних зображеннях із різними геометричними властивостями та на двох реальних зображеннях. За результатами обчислювальних експериментів здійснено порівняння методів за кількістю виявлених ключових точок, якістю їх конфігурації та часом пошуку однієї ключової точки. Встановлено, що жоден із розглянутих методів не є абсолютним переможцем за сукупністю показників, а вибір оптимального методу суттєво залежить від характеристик зображення та вимог конкретної задачі.

Ключові слова: *ключові точки, дескриптор зображення, SIFT, SURF, BRISK, ORB, KAZE, AKAZE, комп'ютерний зір, масштабно-інваріантне перетворення.*

Вступ

Задача пошуку одного зображення на іншому лежить в основі багатьох сучасних систем комп'ютерного зору. Вона є невід'ємною частиною розпізнавання об'єктів і сцен, реконструкції тривимірних структур за серією знімків, побудови стереоскопічних

зображень та відстеження руху в потоці відео [7, 8]. Особливо актуальним є її застосування у задачах автоматичного аналізу аерофотознімків, отриманих із безпілотних літальних апаратів, у системах розпізнавання номерних знаків транспортних засобів та ідентифікації осіб у місцях скупчення людей.

Вирішення задачі зіставлення зображень потребує виокремлення таких ознак, які залишаються стабільними при різних умовах спостереження – зміні масштабу, повороті, зсуві точки огляду або коливаннях освітлення. Такими ознаками є ключові точки – локальні особливості зображення, що добре локалізовані як у просторовій, так і у частотній областях, та характеризуються низькою ймовірністю порушення за рахунок часткового перекриття або шуму.

Розвиток методів виявлення ключових точок розпочався із робіт Г. Моравеца та К. Харріса [4, 12]. Їхні кутові детектори орієнтувались на пошук ділянок зображення з великими градієнтами в усіх напрямках, що добре відповідало потребам стереозіставлення та відстеження руху на коротких відстанях. Однак ці детектори виявились суттєво чутливими до зміни масштабу зображення, що обмежувало їх практичне застосування.

Масштабна інваріантність була досягнута в рамках методу SIFT, запропонованого Д. Лоувом [7, 8]. Метод забезпечує пошук ключових точок у повному просторі масштабів та формує стійкий дескриптор кожної з них на основі локальних градієнтів. Подальший розвиток напряму пов'язаний із розробкою значно швидших методів – SURF [1], BRISK [5], ORB [15], KAZE та AKAZE [2] – що зберігають інваріантні властивості, але суттєво скорочують обчислювальні витрати.

Метою статті є проведення порівняльного аналізу шести сучасних методів виявлення ключових точок на зображеннях – SIFT, SURF, BRISK, ORB, KAZE та AKAZE – за двома ключовими характеристиками: якістю конфігурації знайдених точок та часом пошуку однієї ключової точки. Аналіз проводиться на зображеннях різного типу, що дає змогу оцінити ефективність кожного методу в залежності від змісту зображення.

Виклад основного матеріалу

1. Огляд методів виявлення ключових точок

1.1 Метод SIFT

Метод масштабно-інваріантного перетворення особливостей (Scale-Invariant Feature Transform, SIFT) [7, 8] є теоретично найбільш обґрунтованим серед розглянутих підходів. Його робота базується на чотирьох послідовних етапах.

Виявлення масштабно-просторових екстремумів. Ключові точки виявляються як локальні екстремуми у просторі масштабів, що формується за допомогою різниці Гаусових функцій (Difference of Gaussians, DoG):

$$\begin{aligned} D(x,y,\sigma) &= \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) \circ I(x,y) = \\ &= L(x,y,k\sigma) - L(x,y,\sigma). \end{aligned} \tag{1}$$

де $G(x,y,\sigma)$ – Гаусова функція з параметром масштабу σ, $I(x,y)$ – вхідне зображення, $L(x,y,\sigma)$ – його згортка з Гаусовою функцією. Ця функція є наближенням нормалізованого Лапласіана Гауса $\sigma^2\nabla^2 G$, що забезпечує теоретично коректну масштабну інваріантність.

Кожна точка вибірки порівнюється з вісьмома сусідами на поточному рівні масштабу та з дев'ятьма сусідами на сусідніх рівнях – усього 26 сусідів. Точка

обирається як кандидат лише в тому разі, якщо вона є максимальною або мінімальною серед усіх сусідів.

Точна локалізація ключових точок. Для кожного кандидата виконується підбір тривимірної квадратичної функції до локальних точок вибірки шляхом розкладу в ряд Тейлора:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}, \tag{2}$$

де $\mathbf{x} = (x, y, \sigma^T)$ – зміщення від точки вибірки. Це дозволяє відкидати нестабільні екстремуми з низьким контрастом та точки, розташовані вздовж країв, де матриця Гессіана **H** має велике значення відношення головних кривин $r = \alpha/\beta$ (де $\alpha \geq \beta$ – власні значення **H**), перевіряючи умову:

$$\frac{Tr(\mathbf{H})^2}{|\mathbf{H}|} < \frac{(r+1)^2}{r}. \tag{3}$$

Визначення напрямку. Для кожної ключової точки будується гістограма напрямків локальних градієнтів із 36 інтервалами в діапазоні 0-360°. Домінуючий напрямок призначається точці, що забезпечує інваріантність до повороту зображення.

Формування дескриптора. Навколо кожної ключової точки у відповідному масштабі обчислюються гістограми градієнтів у 16 областях розміром 4×4. Дескриптор являє собою вектор із 128 елементів, нормалізований для досягнення інваріантності до змін освітлення.

1.2 Метод SURF

Метод SURF (Speeded-Up Robust Features) [1] розроблено з метою прискорення обчислень порівняно з SIFT. Він замінює Гаусові фільтри швидкими апроксимаціями у вигляді бокс-фільтрів, що дозволяє ефективно обчислювати наближення детермінанта матриці Гессіана за допомогою інтегральних зображень. Дескриптор SURF будується на основі відгуків вейвлетів Хаара. Метод демонструє значно вищу швидкість при порівнянній якості виявлення ключових точок, хоча може поступатися SIFT при суттєвих змінах освітлення та точки огляду сцени.

1.3 Метод BRISK

Метод BRISK (Binary Robust Invariant Scalable Keypoints) [5] використовує бінарний дескриптор, що формується на основі порівняння яскравостей пікселів у заздалегідь визначеному шаблоні навколо ключової точки. Бінарна природа дескриптора суттєво скорочує обчислювальні витрати на порівняння. Метод є масштабно-інваріантним завдяки застосуванню піраміди зображень. Серед переваг – висока стійкість до змін освітлення та малий час обчислення дескриптора.

1.4 Метод ORB

Метод ORB (Oriented FAST and Rotated BRIEF) [15] поєднує детектор FAST для виявлення ключових точок із модифікованим дескриптором BRIEF, доповненим механізмом врахування орієнтації точки. FAST відзначається надзвичайно низькими обчислювальними витратами на порівняння інтенсивностей пікселів у кільцевому

шаблоні. Дескриптор BRIEF є бінарним, що забезпечує ефективне порівняння. ORB є вільним від патентних обмежень та широко використовується у задачах реального часу.

1.5 Методи KAZE та AKAZE

Методи KAZE та AKAZE [2] використовують нелінійне дифузійне розмиття замість Гаусового, що дозволяє краще зберігати межі об'єктів при побудові масштабного простору. KAZE формує нелінійний дескриптор на основі нелінійно розмитих зображень, а AKAZE використовує бінарний дескриптор M-LDB (Modified Local Difference Binary). Обидва методи є масштабно-інваріантними; AKAZE відрізняється значно вищою швидкодією завдяки бінарному дескриптору.

2. Постановка задачі дослідження

Для порівняння ефективності розглянутих методів виділено дві ключові характеристики:

- кількість виявлених ключових точок як індикатор повноти охоплення деталей зображення;
- середній час пошуку однієї ключової точки (у мілісекундах) як міра обчислювальної ефективності методу.

Важливо наголосити, що поняття «надмірності» та «недостатності» кількості ключових точок є суб'єктивними і залежать від конкретної задачі. При аналізі результатів ураховується не лише кількість точок, а й їх просторова конфігурація відносно значущих деталей зображення – кутів, перетинів, меж об'єктів.

Усі методи реалізовані з використанням бібліотеки комп'ютерного зору OpenCV [20] та мови програмування Python.

3. Тестові зображення та план дослідження

Для дослідження використано чотири синтетичні та два реальні зображення.

Синтетичні зображення навмисно підібрані так, щоб охопити різні геометричні ситуації, характерні для практичних задач аналізу зображень (рис. 1).

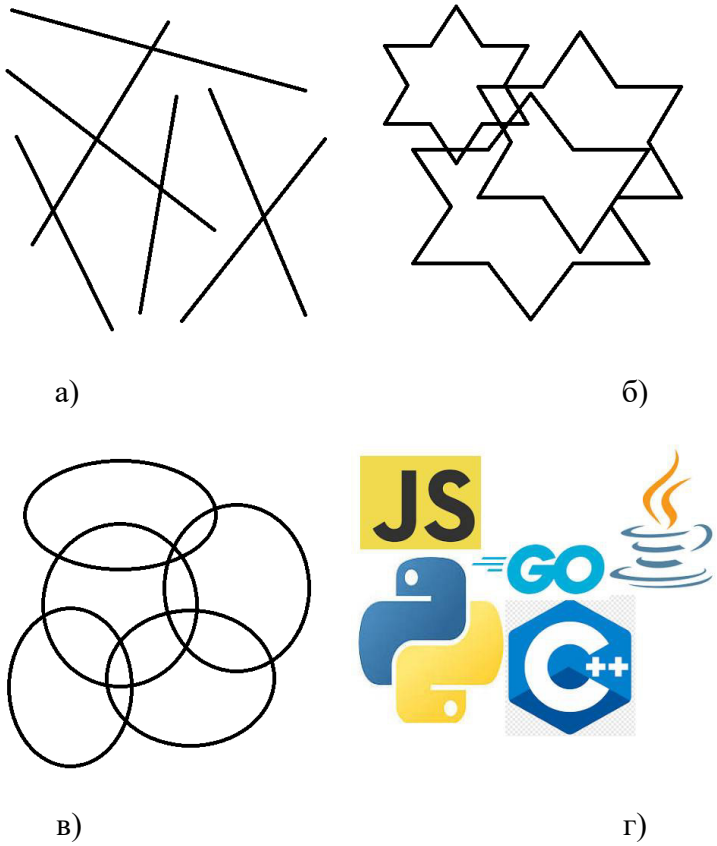
Зображення 1.а складається з відрізків, деякі з яких перетинаються. На ньому лише 5 точок перетину, що дозволяє оцінити поведінку методів на зображеннях із мінімальною кількістю потенційних «точок прив'язки» і виключно прямолінійними структурами.

Зображення 1.б містить замкнені ламані (дві зірки, накладені одна на одну), що утворюють значно більшу кількість кутів і перетинів. Це зображення наближається до ситуацій, у яких людина могла б впевнено ідентифікувати об'єкти.

Зображення 1.в побудоване із трьох еліпсів, що перетинаються. Характерна особливість – відсутність різких кутів: прямих «точок прив'язки» значно менше, ніж на зображенні б), а локальні деталі розподілені по плавним кривим.

Зображення 1.г є кольоровим і містить кілька логотипів сучасних мов програмування, що мають різноманітні кольори, деталі та текстові елементи. Воно найближче до реальних зображень, на яких застосовуються методи пошуку ключових точок.

Реальні тестові зображення є титульними сторінками двох книг з математики та програмування. Ці зображення містять складні кольорові візерунки, різноманітні шрифти та багату деталізацію, що наближує умови тестування до практичних задач комп'ютерного зору.



а) б)

в) г)

Рис. 1. Тестові синтетичні зображення: а) відрізки з перетинами; б) замкнені ламані (зірки); в) еліпси; г) кольорове зображення з логотипами мов програмування.

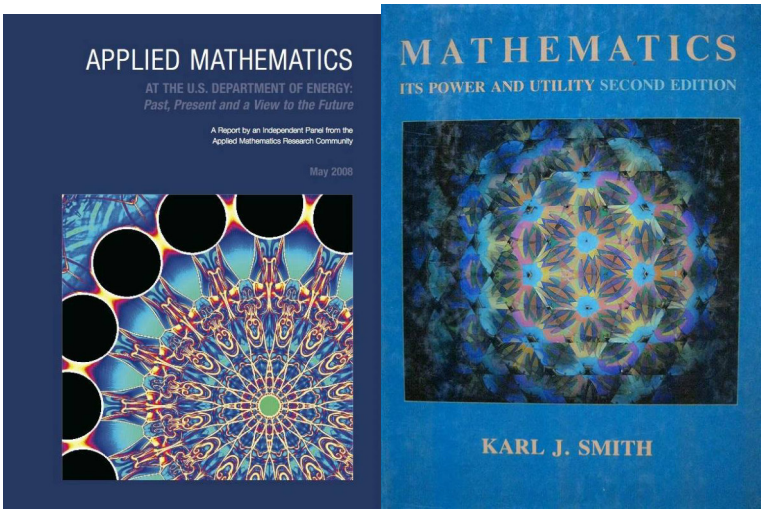


Рис. 2. Тестові реальні зображення (титульні сторінки книг), використані на другому етапі дослідження.

Дослідження проводилось у 2 етапи.

На першому етапі проводилась оцінка виявлення ключових точок на синтетичних зображеннях. Оцінювалась робота кожного методу для кожного зображення. Для цього виконувались наступні кроки:

1) відкривалось зображення та конвертувалось у відтінки сірого кольору;

2) ініціалізувалась реалізація відповідного методу у бібліотеці комп'ютерного зору OpenCV [20];
3) визначались ключові точки з оцінкою часу роботи;
4) дані про ключові точки зберігались у текстовому файлі. Для поточного дослідження ця інформація не потрібна, але вона може знадобитись у продовженні досліджень;
5) обраховувались дескриптори у відповідності до обраного методу та виводилась інформація про них у консоль;
6) отримані дані також зберігались у файл для використання результатів;
7) знайдені ключові точки відображались на зображенні для візуальної оцінки результату роботи методу.

На другому етапі проводилась оцінка роботи методів на реальних зображеннях. Ідеєю цього етапу було використання саме реальних зображень та застосування досліджуваних методів не один раз. Останнє досягалось за рахунок кількох застосувань одного й того ж методу на досліджуваних зображеннях. Для прикладу, ми знаходили на 2-х зображеннях ключові точки по 10 разів, що емулювало пошук ключових точок на 20 зображеннях.

Другий етап проводився з використанням наступних кроків:

1) попередньо усі тестові зображення конвертувались у відтінки сірого кольору та зберігались у окремому списку. Це робилось для того, щоб потім у кожному методі не витрачати час на конвертування;
2) перебирались методи по черзі і для кожного методу задану кількість разів (у програмі – 10) на кожному зображенні шукались ключові точки з визначенням часу роботу. Час додавався до сумарного, кількість знайдених ключових точок теж додавалась до сумарної. Знайдена кількість точок та сумарний час роботи повертались у основну програму;
3) отриманий час роботи, кількість знайдених ключових точок, середній час пошуку однієї точки та назва методу зберігались у результуючий файл для аналізу результатів.

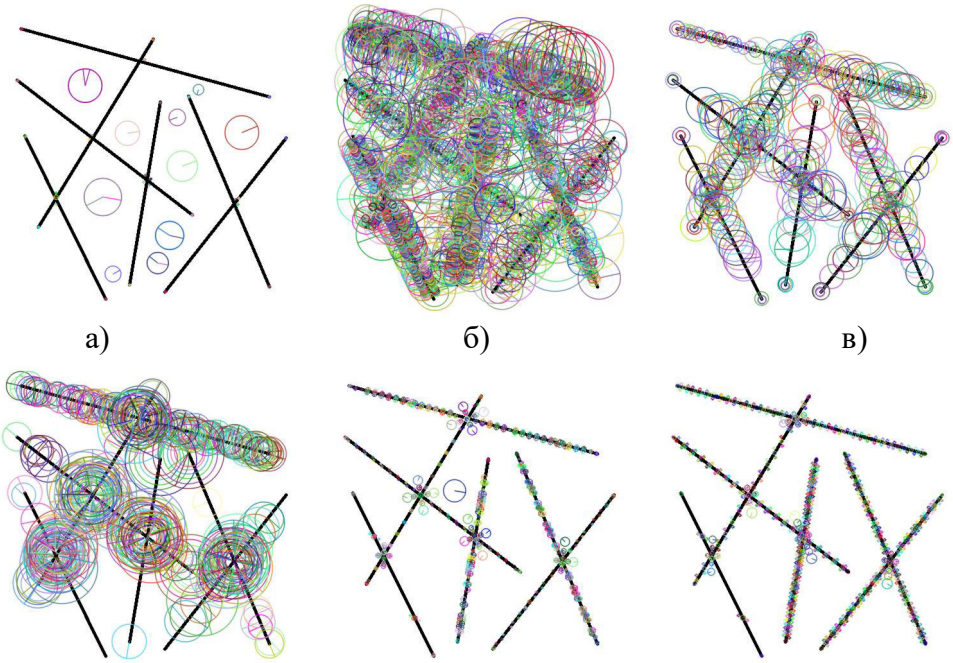
4. Результати дослідження

4.1 Аналіз результатів на синтетичних зображеннях

Зображення з відрізками (рис. 3). На зображенні, що містить прямолінійні відрізки з п'ятьма точками перетину, методи продемонстрували суттєво різні результати.

Метод SIFT виявив найменшу кількість ключових точок, причому розташовані вони поза безпосередньою близькістю до точок перетину відрізків, що з практичної точки зору є небажаним для задачі зіставлення зображень. Методи SURF, BRISK та ORB виявили значно більшу кількість ключових точок; при цьому лише для ORB вони сконцентровані переважно у точках перетину – саме там, де «прив'язка» при зіставленні є найбільш надійною. KAZE та AKAZE розподілили ключові точки вздовж усієї довжини відрізків аналогічно до SURF та BRISK.

З точки зору обчислювальної ефективності методи розташовуються у такому порядку (від найменшого часу на одну точку): BRISK (0.04 мс), SURF (0.06 мс), AKAZE (0.14 мс), ORB (0.20 мс), KAZE (1.25 мс), SIFT (2.5 мс). Таким чином, SIFT є найповільнішим для цього типу зображень.

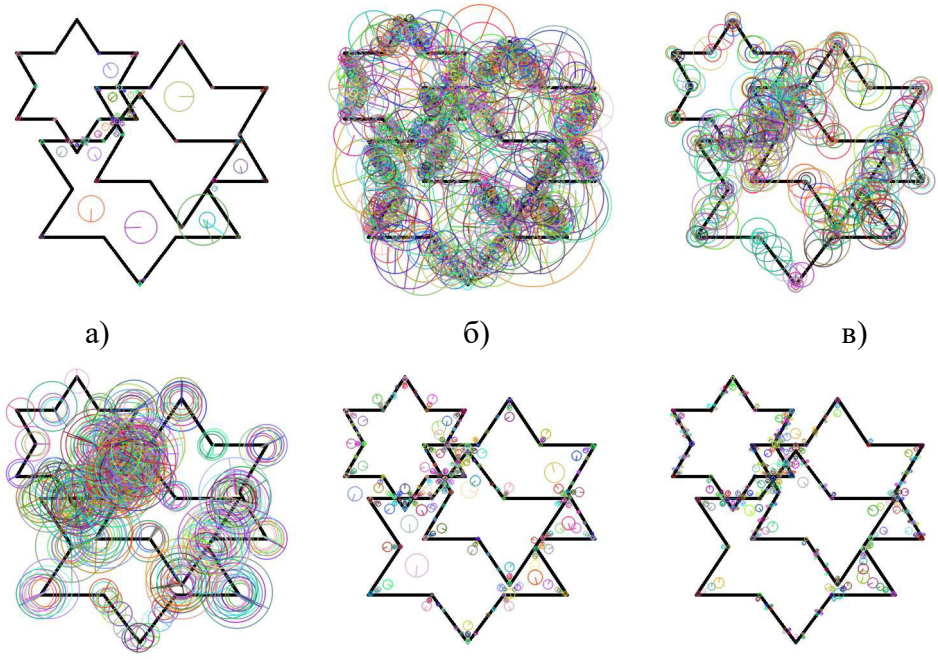


а) б) в)

г) д) е)

Рис. 3. Синтетичне зображення з відрізками із нанесеними ключовими точками: а) SIFT, б) SURF, в) BRISK, г) ORB, д) KAZE, е) AKAZE

Зображення з ламаними (рис. 4). При аналізі зображення з замкненими ламаними (зірками) ситуація дещо змінюється.



а) б) в)

г) д) е)

Рис. 4. Синтетичне зображення з ламаними із нанесеними ключовими точками: а) SIFT, б) SURF, в) BRISK, г) ORB, д) KAZE, е) AKAZE

SIFT та KAZE виявили найменшу кількість точок; більшість точок SIFT розташована поза вершинами та перетинами фігур, тоді як KAZE частково локалізує

точки у значущих місцях. SURF знайшов велику кількість точок по всій довжині ребер. Методи BRISK та ORB переважно концентрують точки у вершинах та перетинах ребер, хоча для обох методів характерна надмірна кількість точок з однаковими координатами та різними напрямками. З огляду на конфігурацію точок при достатньому їх фільтруванні перевагу можна надати методу ORB.

Порядок часової ефективності для цього зображення: BRISK (0.04 мс), SURF (0.09 мс), ORB (0.20 мс), AKAZE (0.30 мс), SIFT (1.06 мс), KAZE (1.70 мс).

Зображення з еліпсами (рис. 5).

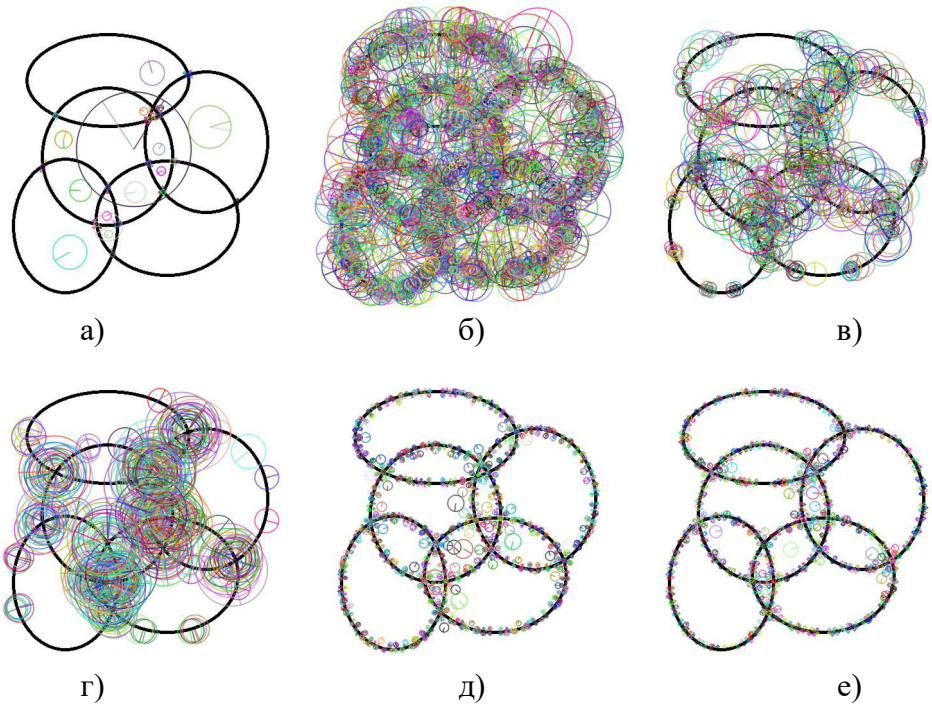


Рис. 5. Синтетичне зображення з еліпсами із нанесеними ключовими точками: а) SIFT, б) SURF, в) BRISK, г) ORB, д) KAZE, е) AKAZE

Загальна картина для зображення з еліпсами є подібною до попереднього випадку. SIFT виявив найменшу кількість точок, переважно у зонах з вираженою кривизною. ORB знову забезпечив раціональну конфігурацію точок у значущих локаціях при прийнятній швидкодії.

Порядок часової ефективності: BRISK (0.03 мс), SURF (0.06 мс), AKAZE (0.11 мс), ORB (0.20 мс), KAZE (0.60 мс), SIFT (1.70 мс).

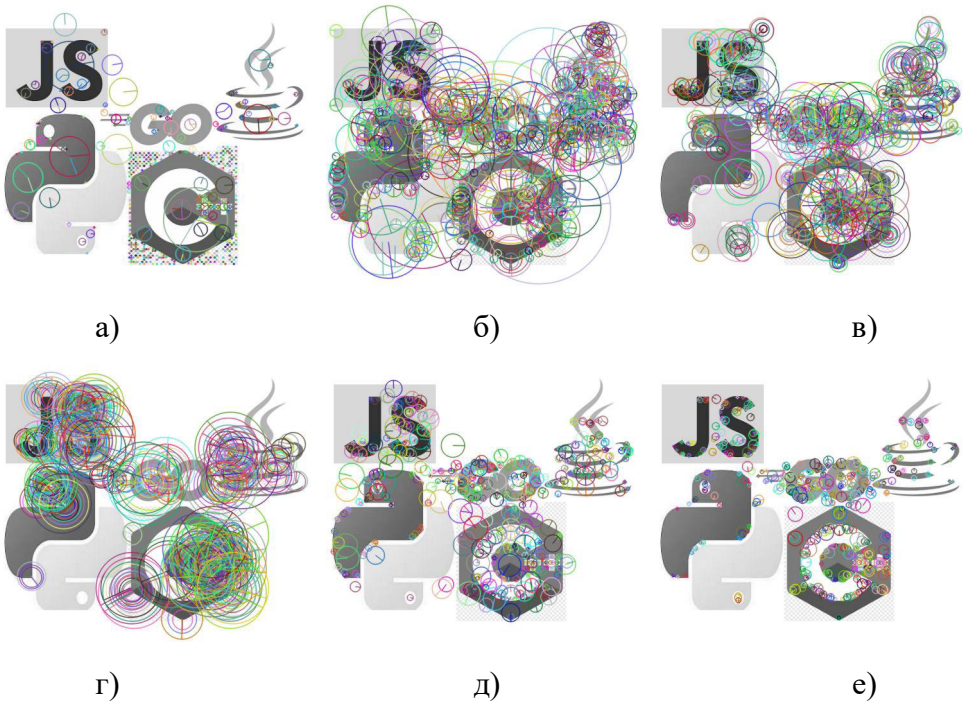
Зображення з логотипами мов програмування (рис. 6).

Кольорове зображення з різноманітним вмістом показало більш збалансовані результати. SIFT виявив помірну кількість точок, розподілених по значущих деталях зображення – кутах літер, межах логотипів. ORB знов забезпечив другий найменший час обчислення (0.18 мс) після BRISK (0.03 мс) при задовільній конфігурації ключових точок.

Порядок часової ефективності: BRISK (0.03 мс), ORB (0.18 мс), AKAZE (0.20 мс), SURF (0.20 мс), SIFT (0.24 мс), KAZE (1.20 мс).

4.2 Результати на реальних зображеннях

На другому етапі дослідження кожен метод застосовувався по 10 разів до кожного з двох реальних зображень (титульних сторінок книг). Зведені результати наведено у таблиці 1.



а) б) в)

г) д) е)

Рис. 6. Синтетичне зображення з логотипами мов програмування із нанесеними ключовими точками: а) SIFT, б) SURF, в) BRISK, г) ORB, д) KAZE, е) AKAZE

Таблиця 1

Результати застосування досліджуваних методів до пошуку ключових точок на реальних зображеннях

Метод	Ключових точок	Час пошуку усіх точок, мс	Час пошуку однієї точки, мс
BRISK	148300	4800	0.0324
ORB	10000	703	0.0703
SURF	86140	9202	0.1068
AKAZE	48650	7215	0.1483
SIFT	63730	14036	0.2202
KAZE	53230	45838	0.8611

Примітка: методи у таблиці впорядковано за зростанням усередненого часу пошуку однієї ключової точки.

Дані таблиці 1 свідчать про суттєву варіативність результатів між методами. BRISK демонструє найменший час пошуку однієї точки (0.0324 мс), однак виявляє надмірно велику їх кількість (148300 за 20 застосувань), що свідчить про низьку вибірковість методу та ймовірну присутність значної частки нестабільних точок. ORB забезпечує в 14.8 рази меншу кількість виявлених точок порівняно з BRISK та другий найменший час обчислення (0.0703 мс). SURF виявляє велику кількість точок (86140), але витрачає значний час на їх обробку. SIFT, попри найбільш теоретично обґрунтований підхід, демонструє час обчислення більш ніж у 6 разів більший, ніж ORB. KAZE є явним аутсайдером за швидкодією – час пошуку однієї точки перевищує аналогічний показник BRISK у 26.6 рази.

5. Аналіз результатів та наукова новизна

5.1 Зіставлення методів за сукупністю показників

Проведений аналіз демонструє принципове протиріччя між якістю виявлення ключових точок та швидкодією методів. Теоретично найбільш обґрунтований метод SIFT, що використовує точну апроксимацію масштабно-нормалізованого Лапласіана Гауса та детальну верифікацію кандидатів, виявляється найповільнішим у більшості тестованих випадків. Натомість методи, що використовують бінарні дескриптори (BRISK, ORB), досягають значного прискорення ціною деякого зниження вибірковості або точності локалізації.

Суттєвим спостереженням є те, що найшвидший метод – BRISK – виявляє надмірну кількість ключових точок, розподілених по зображенню, що ускладнює їх використання у практичних задачах без додаткової фільтрації. ORB демонструє більш збалансований результат: кількість виявлених точок є значно меншою та їх конфігурація краще відповідає значущим локальним деталям зображення.

Варто зазначити, що характер залежності між типом зображення та ефективністю методу є нетривіальним. На зображенні з відрізками (мінімальна кількість значущих деталей) жоден метод, крім ORB, не зосередив точки переважно у точках перетину. На зображеннях із замкненими ламаними та еліпсами методи BRISK та ORB виявились більш «прицільними», тоді як SURF та KAZE/AKAZE рівномірно покрили всі ребра.

5.2 Особливості методів щодо типу контенту

Детальний аналіз результатів дозволяє сформулювати наступні узагальнення.

Для зображень із переважно прямолінійними структурами та малою кількістю характерних деталей (тип а) метод SIFT забезпечує найбільш стриману реакцію, уникаючи масового виявлення нестабільних точок вздовж прямих ліній. Це, у поєднанні з надійністю дескриптора, може бути перевагою у задачах, де точність важливіша за швидкість.

Для зображень із кутовими структурами та великою кількістю перетинів (типи б та в) метод ORB забезпечує прийнятний компроміс між якістю конфігурації точок та швидкодією. BRISK, незважаючи на найменший час обчислення, потребує додаткового кроку фільтрації.

Для насичених кольорових зображень із текстурою та різноманітними деталями (типи г та реальні зображення) відмінності між методами з точки зору якості конфігурації стають менш вираженими, тоді як перевага ORB та BRISK у швидкодії зберігається.

5.3 Практична значущість дослідження

Практична значущість полягає у тому, що отримані результати можуть слугувати основою для обґрунтованого вибору методу виявлення ключових точок під конкретну прикладну задачу з урахуванням вимог до точності, швидкодії та типу оброблюваних зображень.

Висновки

У результаті проведеного дослідження виконано порівняльний аналіз шести методів виявлення ключових точок на зображеннях різного типу. Сформульовано такі висновки.

1. Методи виявлення ключових точок суттєво відрізняються за ефективністю залежно від вмісту зображення. Підхід, що є оптимальним для одного типу зображень, може виявитися неефективним для іншого, тому вибір методу повинен ґрунтуватись на аналізі характеристик задачі.

2. Найвища обчислювальна ефективність (найменший час на одну ключову точку) досягається методами BRISK та ORB, що використовують бінарні дескриптори. Однак BRISK демонструє надмірну кількість виявлених точок, що потребує додаткової фільтрації.

3. Метод ORB забезпечує найбільш збалансований результат: прийнятний час обчислення (0,0703 мс на точку для реальних зображень), поміркована кількість виявлених точок та задовільна їх конфігурація відносно значущих деталей зображення. Це робить ORB перспективним вибором для задач реального часу та систем із обмеженими обчислювальними ресурсами.

4. Метод SIFT, незважаючи на найбільші обчислювальні витрати, забезпечує стабільну та теоретично обґрунтовану поведінку. Для задач, де точність є пріоритетом над швидкодією, SIFT залишається конкурентоспроможним.

5. Метод KAZE є найповільнішим серед розглянутих (0,8611 мс на точку), що суттєво обмежує його застосування у задачах, де важлива швидкодія.

6. Встановлено, що не існує методу, який є абсолютним переможцем за сукупністю характеристик. Ефективне застосування методів пошуку ключових точок потребує знання особливостей їх роботи на різних типах зображень, наявних обчислювальних ресурсів та конкретних вимог задачі.

Список використаної літератури

1. Bay H., Ess A., Tuytelaars T., Van Gool L. Speeded-Up Robust Features (SURF) // Computer Vision and Image Understanding. – 2008. – Vol. 110, № 3. – P. 346–359.
2. Brown M., Lowe D.G. Invariant features from interest point groups // Proceedings of the British Machine Vision Conference. – Cardiff, Wales, 2002. – P. 656–665.
3. Crowley J. L., Parker A. C. A representation for shape based on peaks and ridges in the difference of low-pass transform // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1984. – № 6(2). – P. 156–170.
4. Harris C., Stephens M. A combined corner and edge detector // Proceedings of the Fourth Alvey Vision Conference. – Manchester, UK, 1988. – P. 147–151.
5. Leutenegger S., Chli M., Siegwart R. Y. BRISK: Binary Robust Invariant Scalable Keypoints // Proceedings of the IEEE International Conference on Computer Vision. – 2011. – P. 2548–2555.
6. Lindeberg T. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention // International Journal of Computer Vision. – 1993. – № 11(3). – P. 283–318.
7. Lowe D. G. Object recognition from local scale-invariant features // Proceedings of the International Conference on Computer Vision. – 1999. – P. 1150–1157.
8. Lowe D. G. Distinctive image features from scale-invariant keypoints // International Journal of Computer Vision. – 2004. – Vol. 60, № 2. – P. 91–110.
9. Matas J., Chum O., Urban M., Pajdla T. Robust wide baseline stereo from maximally stable extremal regions // Proceedings of the British Machine Vision Conference. – Cardiff, Wales, 2002. – P. 384–393.
10. Mikolajczyk K., Schmid C. An affine invariant interest point detector // Proceedings of the European Conference on Computer Vision. – Copenhagen, Denmark, 2002. – P. 128–142.
11. Mikolajczyk K., Zisserman A., Schmid C. Shape recognition with edge-based features // Proceedings of the British Machine Vision Conference. – Norwich, UK, 2003.
12. Moravec H. Rover visual obstacle avoidance // Proceedings of the International Joint Conference on Artificial Intelligence. – Vancouver, Canada, 1981. – P. 785–790.
13. Nelson R. C., Selinger A. Large-scale tests of a keyed, appearance-based 3-D object recognition system // Vision Research. – 1998. – № 38(15). – P. 2469–2488.
14. Pope A. R., Lowe D. G. Probabilistic models of appearance for 3-D object recognition // International Journal of Computer Vision. – 2000. – № 40(2). – P. 149–167.
15. Rublee E., Rabaud V., Konolige K., Bradski G. ORB: an efficient alternative to SIFT or SURF // Proceedings of the IEEE International Conference on Computer Vision. – 2011. – P. 2564–2571.

16. Schmid C., Mohr R. Local grayvalue invariants for image retrieval // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1997. – № 19(5). – P. 530–534.
17. Shokoufandeh A., Marsic I., Dickinson S. J. View-based object recognition using saliency maps // Image and Vision Computing. – 1999. – № 17. – P. 445–460.
18. Torr P. Motion Segmentation and Outlier Detection: Ph.D. Thesis / University of Oxford. – Oxford, UK, 1995.
19. Zhang Z., Deriche R., Faugeras O., Luong Q. T. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry // Artificial Intelligence. – 1995. – Vol. 78. – P. 87–119.
20. Бібліотека комп'ютерного зору OpenCV [Електронний ресурс]. – Режим доступу: https://opencv.org/

References

1. Bay H., Ess A., Tuytelaars T., Van Gool L. (2008) Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding, 110(3), 346–359.
2. Brown M., Lowe D.G. (2002) Invariant features from interest point groups. Proceedings of the British Machine Vision Conference, Cardiff, Wales, 656–665.
3. Crowley J.L., Parker A.C. (1984) A representation for shape based on peaks and ridges in the difference of low-pass transform. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(2), 156–170.
4. Harris C., Stephens M. (1988) A combined corner and edge detector. Proceedings of the Fourth Alvey Vision Conference, Manchester, UK, 147–151.
5. Leutenegger S., Chli M., Siegwart R.Y. (2011) BRISK: Binary Robust Invariant Scalable Keypoints. Proceedings of the IEEE International Conference on Computer Vision, 2548–2555.
6. Lindeberg T. (1993) Detecting salient blob-like image structures and their scales with a scale-space primal sketch. International Journal of Computer Vision, 11(3), 283–318.
7. Lowe D.G. (1999) Object recognition from local scale-invariant features. Proceedings of the International Conference on Computer Vision, 1150–1157.
8. Lowe D.G. (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110.
9. Matas J., Chum O., Urban M., Pajdla T. (2002) Robust wide baseline stereo from maximally stable extremal regions. Proceedings of the British Machine Vision Conference, Cardiff, Wales, 384–393.
10. Mikolajczyk K., Schmid C. (2002) An affine invariant interest point detector. Proceedings of the European Conference on Computer Vision, Copenhagen, Denmark, 128–142.
11. Mikolajczyk K., Zisserman A., Schmid C. (2003) Shape recognition with edge-based features. Proceedings of the British Machine Vision Conference, Norwich, UK.
12. Moravec H. (1981) Rover visual obstacle avoidance. Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver, Canada, 785–790.
13. Nelson R.C., Selinger A. (1998) Large-scale tests of a keyed, appearance-based 3-D object recognition system. Vision Research, 38(15), 2469–2488.
14. Pope A.R., Lowe D.G. (2000) Probabilistic models of appearance for 3-D object recognition. International Journal of Computer Vision, 40(2), 149–167.
15. Rublee E., Rabaud V., Konolige K., Bradski G. (2011) ORB: an efficient alternative to SIFT or SURF. Proceedings of the IEEE International Conference on Computer Vision, 2564–2571.
16. Schmid C., Mohr R. (1997) Local grayvalue invariants for image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(5), 530–534.
17. Shokoufandeh A., Marsic I., Dickinson S.J. (1999) View-based object recognition using saliency maps. Image and Vision Computing, 17, 445–460.
18. Torr P. (1995) Motion Segmentation and Outlier Detection. Ph.D. Thesis, University of Oxford, UK.
19. Zhang Z., Deriche R., Faugeras O., Luong Q.T. (1995) A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Artificial Intelligence, 78, 87–119.
20. OpenCV: Open Source Computer Vision Library. Available at: https://opencv.org/

HAVRYUSHENKO Anna Mykolaivna,
teacher of Mathematics and Computer Science at Cherkasy General Education School of Levels I-III No. 32 of Cherkasy City Council, Cherkasy, Ukraine
BOROZDYKH Kateryna Serhiivna,
student of Cherkasy General Education School of Levels I–III No. 32 of Cherkasy City Council, Cherkasy, Ukraine

COMPARATIVE ANALYSIS OF KEYPOINT DETECTION METHODS IN IMAGES

Summary. Introduction. *Image search – finding one image within another – is a fundamental problem in computer vision, with applications in object recognition, motion tracking, 3D reconstruction, aerial image analysis and autonomous navigation systems. The core challenge is identifying local image features, called keypoints, that remain stable under changes in scale, rotation, viewpoint and illumination. Since the pioneering work of Moravec and Harris on corner detection, and the subsequent development of SIFT by Lowe, the field has produced a number of competing methods that vary significantly in their theoretical foundations, accuracy and computational efficiency.*

Purpose. *This paper presents a systematic comparison of six keypoint detection methods – SIFT, SURF, BRISK, ORB, KAZE and AKAZE – evaluated on images of varying geometric complexity. Two evaluation criteria are used: the spatial configuration quality of detected keypoints relative to salient image features, and the average computation time per keypoint.*

Results. *The mathematical background of SIFT is described in detail, covering scale-space extrema detection via the Difference of Gaussians function, precise keypoint localization using Taylor series expansion of the scale-space function, orientation assignment and descriptor computation. For all six methods, experiments were conducted on four synthetic images (line segments, closed polylines, ellipses, and a color image with programming language logos) and two real-world images (book covers). Results show that BRISK and ORB are the fastest methods, with per-keypoint times of 0.032 ms and 0.070 ms respectively on real images. However, BRISK detects an excessive number of keypoints (148,300 over 20 trials), many of which are unstable. ORB provides a more balanced result: moderate keypoint count (10,000), the second-fastest computation time, and a spatial configuration concentrated near meaningful image features such as corners and intersections. SIFT, while the slowest method (0.220 ms per keypoint), provides a theoretically well-grounded and selective response. KAZE is the slowest across all test cases (0.861 ms per keypoint), limiting its practical use in time-critical applications.*

Conclusion. *No single method dominates across all evaluation criteria and image types. The optimal choice depends on the characteristics of the target images and the specific requirements of the application. For real-time systems with limited computational resources, ORB provides the best trade-off between speed and keypoint quality. For tasks where accuracy is prioritized over speed, SIFT remains competitive. These findings highlight the importance of method selection guided by systematic benchmarking rather than general assumptions.*

Keywords: *keypoints, image descriptor, SIFT, SURF, BRISK, ORB, KAZE, AKAZE, computer vision, scale-invariant feature transform.*

Одержано редакцією 09.11.2024 р.
Прийнято до публікації 11.12.2024 р.

УДК 519.1:004.42

DOI 10.31651/2076-5886-2024-1-34-45

PACS 02.10.Ox

КУРИЛО Олександр Миколайович
директор, учитель інформатики Золотоніської загальноосвітньої школи I–III ступенів № 5

НІЦАК Владислав Костянтинович
учень Золотоніської загальноосвітньої школи I–III ступенів № 5

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ГЕНЕРАЦІЇ КОМБІНАТОРНИХ ОБ'ЄКТІВ

У статті розглянуто основні комбінаторні об'єкти – перестановки, розміщення та комбінації, а також їхні варіанти з повторенням. Наведено математичне обґрунтування відповідних формул підрахунку кількості об'єктів кожного типу. Для кожного з розглянутих видів комбінаторних об'єктів описано та реалізовано алгоритми генерації усіх можливих варіантів. Реалізацію виконано мовою програмування Python з використанням рекурсивного та ітераційного підходів. Особливу увагу приділено порівнянню обох підходів з огляду на їхню ефективність та зручність реалізації. Розроблено програмний продукт із графічним інтерфейсом, що дозволяє наочно демонструвати результати роботи алгоритмів і може застосовуватись як навчальний засіб у курсах математики та інформатики.

Ключові слова: *комбінаторика, перестановки, розміщення, комбінації, рекурсивний алгоритм, Python, генератор.*

Вступ

Комбінаторика є одним із фундаментальних розділів дискретної математики, що займається підрахунком кількості об'єктів, які задовольняють певному набору властивостей, а також побудовою цих об'єктів. Термін «комбінаторика» з'явився ще у 1666 році в роботі Лейбніца, однак у сучасному значенні цей розділ математики інтенсивно розвинувся лише у XX столітті у зв'язку з потребами криптографії, теорії кодування та теорії алгоритмів [4, 6].

Комбінаторні об'єкти – перестановки, розміщення та комбінації – є базовими поняттями, що широко застосовуються у теорії ймовірностей, математичній статистиці, оптимізації та криптографії. Водночас задачі генерації (тобто побудови і виведення) усіх можливих комбінаторних конфігурацій належать до класу NP-задач, де кількість об'єктів зростає надзвичайно швидко зі збільшенням вхідних параметрів. Дослідження ефективних алгоритмів генерації таких об'єктів залишається актуальною задачею, особливо з огляду на потреби навчання та ілюстрації математичних понять.

Незважаючи на те що теоретичний апарат комбінаторики є добре розробленим, практична реалізація відповідних алгоритмів з акцентом на порівнянні рекурсивного та ітераційного підходів, а також на застосуванні сучасних засобів мови програмування Python (зокрема, генераторів та ключового слова yield), залишається методично цінною темою для досліджень прикладного рівня.

Метою статті є систематизація математичних основ основних комбінаторних об'єктів, опис та порівняння алгоритмів їх генерації, реалізованих мовою Python з використанням рекурсивного та ітераційного підходів, а також розробка програмного продукту з графічним інтерфейсом для наочної демонстрації результатів.

Виклад основного матеріалу

1. Огляд задач перебору та класи складності P і NP

Задачі в програмуванні поділяють на класи залежно від часової складності їхнього розв'язання. Задачі класу **P** розв'язуються за поліноміальний час відносно розміру вхідних даних – тобто існує алгоритм зі складністю O(n^k) для деякого фіксованого k. До таких задач належать, зокрема, множення матриць зі складністю $O(n^3)$, сортування масивів та знаходження ейлерового циклу у графі [1, 7].

Задачі класу **NP** (недетерміновано поліноміальні) – це такі, розв'язок яких можна перевірити за поліноміальний час, проте ефективного алгоритму побудови розв'язку для них не знайдено. Прикладами є задача комівояжера та задача визначення існування цілочисельного розв'язку системи лінійних нерівностей [5]. Задачі генерації комбінаторних об'єктів фактично є NP-задачами: кількість об'єктів зростає факторіально або степенево, що робить повний перебір прийнятним лише для невеликих вхідних розмірів.

Важливим відкритим питанням теоретичної інформатики залишається рівність або нерівність класів P і NP. У разі P = NP для всіх NP-задач існував би ефективний алгоритм класу P, що докорінно змінило б підходи до розв'язання задач оптимізації та криптографії. Сьогодні переважає думка, що P ≠ NP, проте формального доведення досі немає [1].

2. Комбінаторні об'єкти: математичні основи

2.1 Правила суми та добутку

В основі комбінаторики лежать два фундаментальні принципи.

Правило суми: якщо об'єкт типу A_1 можна обрати n_1 способами, а об'єкт типу A_2 – n_2 іншими способами (і ці вибори взаємно виключають один одного), то вибір одного з них здійснюється $n_1 + n_2$ способами [6, 8].

Правило добутку: якщо об'єкт A_1 обирається n_1 способами, і після кожного такого вибору об'єкт A_2 може бути обраний n_2 способами, то послідовний вибір обох об'єктів здійснюється $n_1 \cdot n_2$ способами [6, 8].

Ці два правила є підґрунтям для виведення формул підрахунку перестановок, розміщень і комбінацій.

2.2 Перестановки без повторення

Перестановкою з n елементів називається будь-яке впорядковане розташування усіх n елементів. Кількість перестановок розраховується за формулою:

$$P_n = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1 = n!. \qquad (1)$$

Приклад: кількість способів впорядкувати три літери *A*, *B*, *C* дорівнює 3! = 6 (рис. 1):

ABC, *ACB*, *BAC*, *BCA*, *CAB*, *CBA*

2.3 Розміщення без повторення

Розміщенням з n елементів по m називається будь-яка впорядкована підмножина з m елементів n-елементної множини, де $m \le n$. Кількість розміщень:

$$A_n^m = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-m+1) = \frac{n!}{(n-m)!}. \qquad (2)$$

Відмінність від перестановок полягає в тому, що кількість позицій для розміщення менша за загальну кількість елементів (рис. 2).

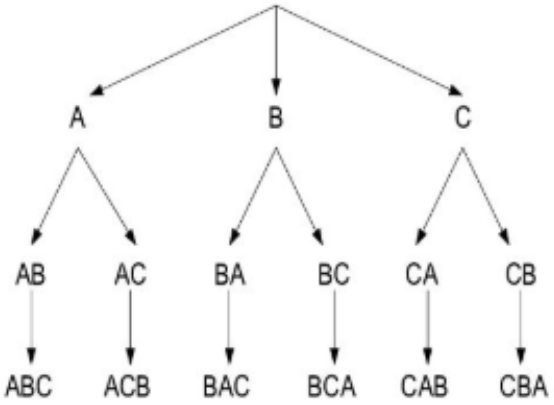


Рис.1. Утворення усіх перестановок 3-х літер: *A*, *B*, *C*

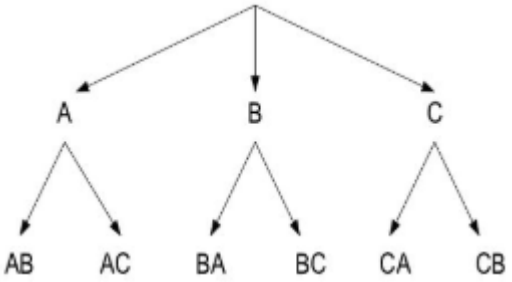


Рис. 2. Утворення усіх розміщень з 3-х літер, *A*, *B*, *C*, по 2

2.4 Комбінації без повторення

Комбінацією з n елементів по m називається підмножина з m елементів, де порядок розташування елементів не важливий. Кількість комбінацій:

$$C_n^m = \frac{A_n^m}{P_m} = \frac{n!}{m!(n-m)!}. \tag{3}$$

Зазначимо, що C_n^m збігається з біноміальним коефіцієнтом у розкладі біному Ньютона:

$$(x+a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k}. \tag{4}$$

Для обчислення біноміальних коефіцієнтів зручно використовувати трикутник Паскаля (рис. 3), де кожний елемент є сумою двох сусідніх елементів рядка вище:

$$C_n^m = C_{n-1}^{m-1} + C_{n-1}^m. \tag{5}$$

2.5 Перестановки з повторенням

Якщо серед n елементів є групи однакових, де кожен елемент i-го типу повторюється k_1 разів ($k_1 + k_2 + ... + k_l = n$), кількість різних перестановок:

$$\overline{P}_n(k_1,\ldots,k_l)=\frac{n!}{k_1!\cdot\ldots\cdot k_l!}. \qquad (6)$$

$$\begin{array}{ccccccccccc}
 & & & & C_0^0 & & & & & & \\
 & & & & C_1^0 & & C_1^1 & & & & \\
 & & & C_2^0 & & C_2^1 & & C_2^2 & & & \\
 & & C_3^0 & & C_3^1 & & C_3^2 & & C_3^3 & & \\
 & C_4^0 & & C_4^1 & & C_4^2 & & C_4^3 & & C_4^4 & \\
. & & . & & . & & . & & . & & . \\
C_n^0 & & C_n^1 & & C_n^2 & \ldots & C_n^{n-2} & & C_n^{n-1} & & C_n^n
\end{array}$$

Рис. 3. Обчислення біноміальних коефіцієнтів за допомогою трикутника Паскаля

Приклад: кількість різних слів зі слова «математика» (10 літер: «м» – 2, «а» – 3, «т» – 2, «е» – 1, «и» – 1, «к» – 1) дорівнює $\frac{10!}{2!\cdot 3!\cdot 2!\cdot 1!\cdot 1!\cdot 1!}$ = 151 200.

2.6 Розміщення з повторенням

Якщо на кожну з k позицій незалежно може бути поставлений будь-який із n елементів (елементи повторюватись можуть), загальна кількість варіантів:

$$\overline{A}_n^k = n^k. \qquad (7)$$

Приклад: кількість кодів чотиризначного цифрового замка – 10^4 = 10000.

2.7 Комбінації з повторенням

Якщо з n типів елементів обирається k елементів із можливим повторенням і порядок не важливий, кількість таких комбінацій:

$$\overline{C}_n^k = C_{k+n-1}^k = \frac{(k+n-1)!}{(n-1)!\cdot k!}. \qquad (8)$$

Для інтуїтивного розуміння цієї формули використовують графічний метод «шарів і перегородок» (рис. 4-5): будь-яке розміщення k однакових шарів у n ящиках однозначно кодується послідовністю з k нулів та (n–1) одиниць – перегородок між ящиками [8].

Рис. 4. Початкові дані до прикладу

Рис. 5. Перегородки задають три ящики, що містять відповідно 4, 1 і 2 об’єкти

2.8 Узагальнена схема методів

Для зручності вибору формули у конкретній задачі можна використати схему-класифікатор, де основними критеріями є: (1) чи важливий порядок елементів у наборі, та (2) чи допускається повторення елементів.

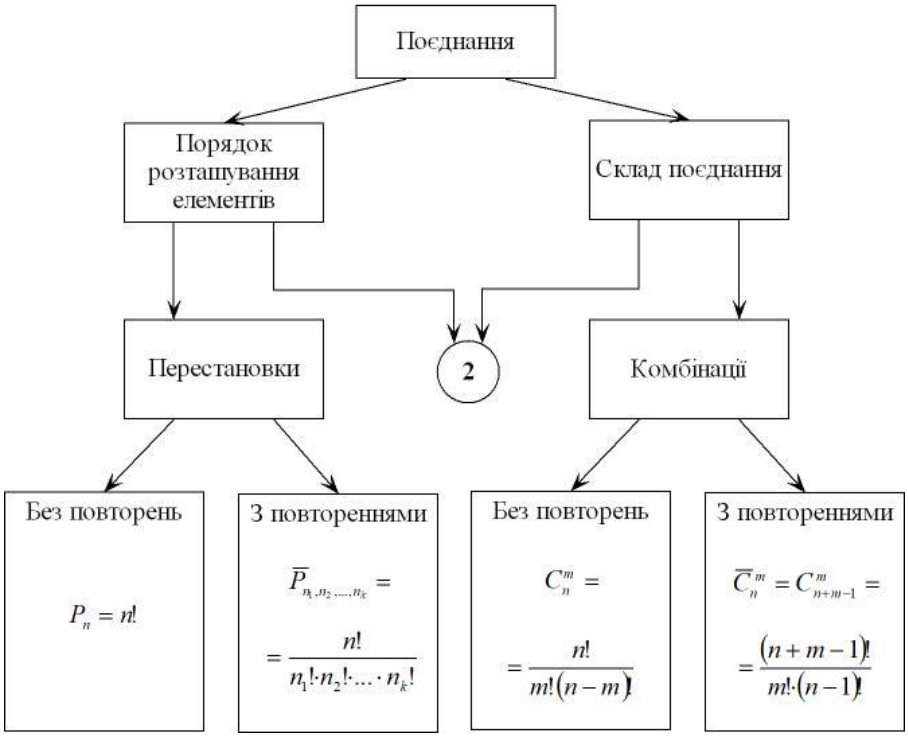


Рис. 6. Формули розрахунку кількості перестановок та комбінацій

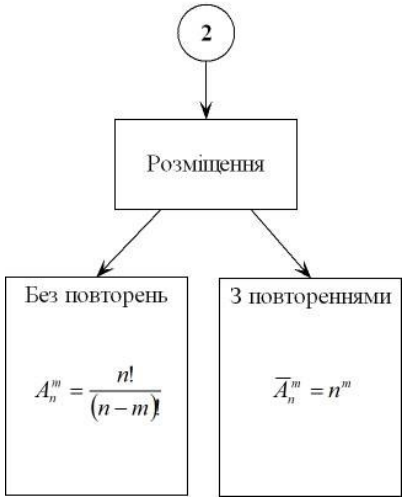


Рис. 7. Формули розрахунку кількості розміщень (продовження рис. 6)

Таблиця 1 узагальнює відповідні формули.

Таблиця 1

Формули підрахунку кількості комбінаторних об'єктів

Об'єкт	Без повторень	З повторенням
Перестановки	$P_n = n!$	$\overline{P} = \frac{n!}{k_1!\cdot\ldots\cdot k_l!}$
Розміщення	$A_n^m = \frac{n!}{(n-m)!}$	$\overline{A}_n^k = n^k$
Комбінації	$C_n^m = \frac{n!}{m!(n-m)!}$	$\overline{C}_n^k = \frac{(k+n-1)!}{(n-1)!\cdot k!}$

3. Алгоритми генерації комбінаторних об'єктів

Для практичної реалізації алгоритмів обрано мову програмування Python. Вибір зумовлений такими характеристиками мови: чіткий і виразний синтаксис, розвинена стандартна бібліотека, підтримка різних парадигм програмування (зокрема, функційної та об'єктно-орієнтованої), крос-платформеність [3, 13].

3.1 Рекурсивний та ітераційний підходи

Ітераційний підхід ґрунтується на циклічному виконанні певної послідовності дій. Аналіз складності таких алгоритмів зводиться до підрахунку ресурсоємності циклічних конструкцій.

Рекурсивний підхід – це такий спосіб організації обчислень, при якому функція звертається до самої себе для розв'язання задачі меншого розміру. Звернення до себе у ході виконання програми називають *прямим ходом рекурсії*, а послідовне повернення з «найглибшого» рекурсивного виклику – *зворотнім ходом* [1].

Рекурсивні алгоритми є природними для задач, де розв'язання зводиться до підзадач того ж типу. Однак вони, як правило, споживають більше оперативної пам'яті через накопичення стеку викликів і можуть бути перетворені на ітераційні.

Класичним прикладом рекурсії є обчислення факторіалу:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

3.2 Генератори та ключове слово yield у Python

Для ефективної генерації великих наборів комбінаторних об'єктів у Python доцільно використовувати *генератори* – спеціальні функції, що повертають значення по одному, не завантажуючи усі результати у пам'ять одночасно.

Ключове слово `yield` відрізняється від `return` тим, що після повернення значення виконання функції не завершується – при наступному зверненні воно продовжується з місця, де було зупинено. Це дозволяє реалізовувати потокову генерацію об'єктів без значних витрат пам'яті – суттєва перевага у задачах комбінаторики, де кількість об'єктів може бути дуже великою [13].

Для порівняння: звичайний список зберігає усі значення в оперативній пам'яті, тоді як генератор повертає їх по одному у міру потреби:

```
# список – зберігає усі значення в пам'яті
list_result = [x for x in range(10)]
# генератор – обчислює по одному
gen_result = (x for x in range(10))
```

3.3 Алгоритм генерації перестановок без повторення

Рекурсивний алгоритм генерує перестановки, послідовно додаючи до поточного префіксу `pref` черговий невикористаний елемент:

```
def recursion_permutations(N, M=None, pref=[]):
    M = N if M is None else M
    if M == 0:
        print(*pref, end=" ", sep="")
        return
    for number in range(1, N + 1):
```

```
        if number in pref:
            continue
        pref.append(number)
        recursion_permutations(N, M - 1, pref)
        pref.pop()
```

Логіка алгоритму: обираємо один елемент із доступних *N*, ставимо його на перше місце і рекурсивно генеруємо перестановки решти елементів. Параметр *M* вказує, скільки позицій залишилось заповнити. При *M* = *N* алгоритм генерує усі перестановки; при *M* < *N* – усі розміщення з *N* по *M* (рис. 8).

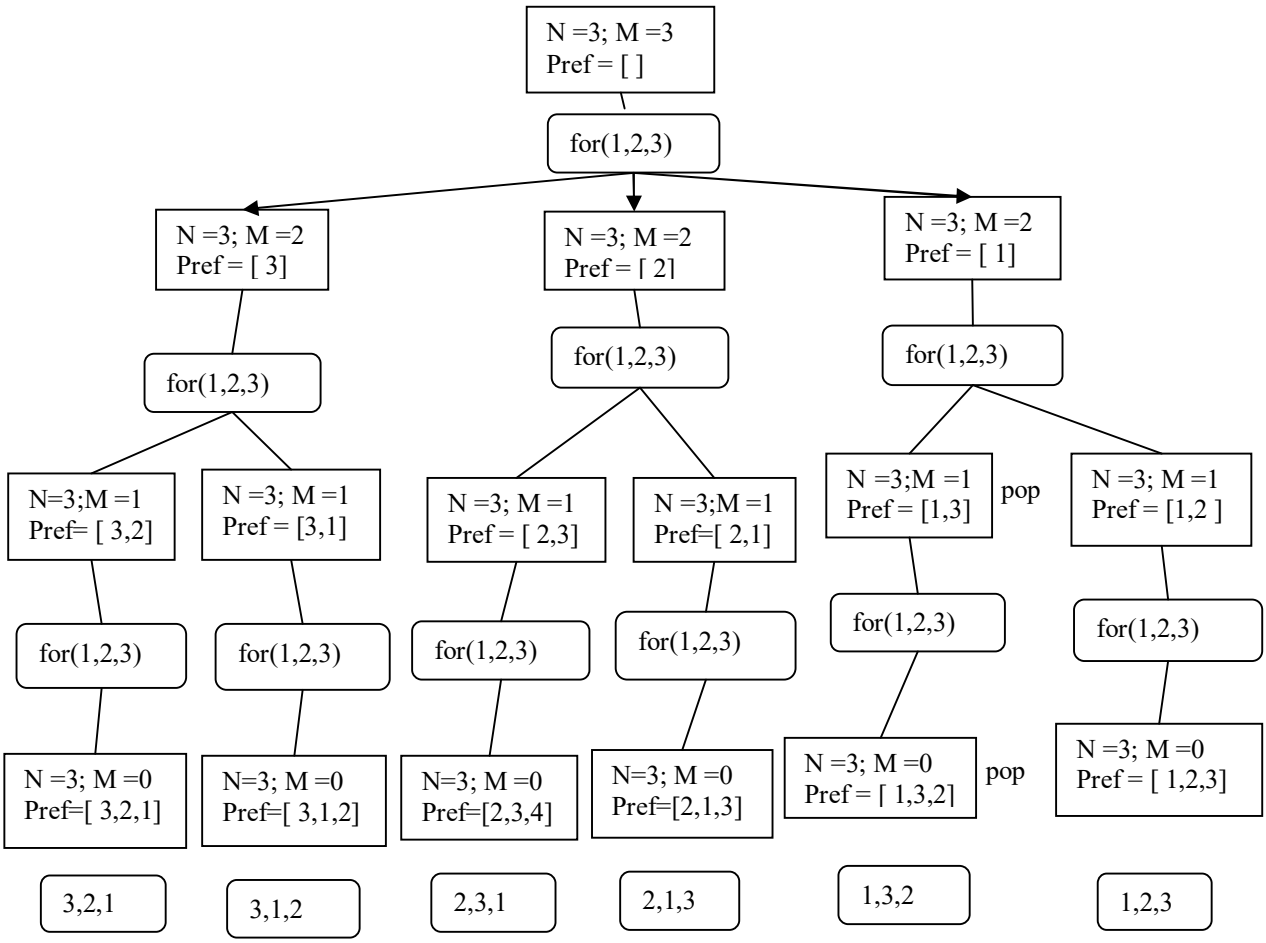


Рис. 8. Схема роботи алгоритму генерації перестановок для вхідних даних *N*=3

3.4 Алгоритм генерації розміщень без повторення

Оскільки розміщення з *N* по *M* відрізняються від перестановок лише тим, що кількість позицій *M* менша за *N*, достатньо передати попередньому алгоритму відповідне значення *M*. Таким чином, один рекурсивний алгоритм охоплює обидва випадки.

3.5 Алгоритм генерації комбінацій без повторення

Для генерації комбінацій реалізовано ітераційний підхід із використанням масиву індексів. Алгоритм починає з першої комбінації (елементи з індексами 0, 1, …, *r*-1) та послідовно переходить до наступної, збільшуючи індекси справа наліво:

```
def iter_combinations(iterable, r):
```

```
    pool = tuple(iterable)
    n = len(pool)
    if r > n:
        return
    indices = list(range(r))
    yield tuple(pool[i] for i in indices)
    while True:
        for i in reversed(range(r)):
            if indices[i] != i + n - r:
                break
        else:
            return
        indices[i] += 1
        for j in range(i + 1, r):
            indices[j] = indices[j - 1] + 1
        yield tuple(pool[i] for i in indices)
```

Конструкція `for ... else` у Python є нестандартною: гілка `else` виконується лише тоді, коли цикл завершився без виклику `break`, тобто коли всі індекси досягли своїх максимальних значень – це сигнал, що усі комбінації вже згенеровано [13].

Також реалізовано рекурсивний варіант алгоритму з використанням `yield`. Незважаючи на те що він демонструє рекурсивний підхід, він менш ефективний через зайві «холості» рекурсивні виклики і наведений передусім для ілюстрації принципу.

3.6 Алгоритми генерації об'єктів із повторенням

Перестановки з повторенням. Оскільки алгоритм перестановок обробляє кожен елемент як унікальний, для отримання перестановок з повторенням достатньо передати список із повторюваними елементами та видалити дублікати з результату. У Python це реалізується за допомогою множин:

```
def permutation_with_repeat(lst):
    if len(lst) == 0:
        return []
    if len(lst) == 1:
        return [lst]
    result = []
    for i in range(len(lst)):
        m = lst[i]
        rem = lst[:i] + lst[i + 1:]
        for p in permutation_with_repeat(rem):
            result.append([m] + p)
    return result
# Видалення дублікатів
res = set([str(x) for x in permutation_with_repeat(lst)])
```

Розміщення з повторенням. Реалізуються ітераційним алгоритмом, що будує усі декартові добутки вхідного набору на себе *k* разів:

```
def placements_with_repeat(*args, repeat=1):
    pools = [tuple(pool) for pool in args] * repeat
    result = [[]]
    for pool in pools:
        result = [x + [y] for x in result for y in pool]
    for prod in result:
```

```
        yield tuple(prod)
```

Комбінації з повторенням. Алгоритм структурно схожий на ітераційний алгоритм комбінацій, але умова зупинки та спосіб оновлення індексів змінені: індекс може повторюватись (кожен наступний індекс ≥ поточному), що відповідає можливості повторного вибору елементів [13].

4. Результати та аналіз

4.1 Порівняння рекурсивного та ітераційного підходів

Обидва підходи дають коректні результати, проте мають різні характеристики (табл. 2).

Таблиця 2

Порівняння рекурсивного та ітераційного підходів

Критерій	Рекурсивний підхід	Ітераційний підхід
Зрозумілість коду	Висока (природна декомпозиція)	Середня (потрібно відстежувати індекси)
Витрати пам'яті	Вищі (стек викликів)	Нижчі
Швидкодія	Нижча (накладні витрати викликів)	Вища
Придатність до великих N	Обмежена глибиною стеку	Краща
Підтримка yield	Так (через рекурсивний генератор)	Так (природна)

Для педагогічних цілей рекурсивний підхід є кращим, оскільки код наочно відображає математичне визначення об'єкта. Для практичних задач із великими вхідними даними перевагу слід надавати ітераційному підходу або генераторам.

4.2 Порівняння кількості об'єктів різних типів

Для ілюстрації швидкості зростання кількості комбінаторних об'єктів наведемо значення для деяких параметрів.

З таблиці 3 видно, що кількість об'єктів зростає надзвичайно швидко. Це підтверджує практичну обмеженість алгоритмів повного перебору і пояснює належність задач генерації до класу NP.

Таблиця 3

Кількість комбінаторних об'єктів залежно від параметрів

N	M	Перестановки P_n	Розміщення A_n^m	Комбінації C_n^m
3	2	6	6	3
4	2	24	12	6
5	3	120	60	10
6	3	720	120	20
7	4	5040	840	35
10	5	3628800	30240	252

4.3 Програмний продукт

Реалізований програмний продукт має графічний інтерфейс, побудований із використанням бібліотеки PyQt5 (обгортка PySide2) [2]. Інтерфейс поділено на чотири вертикальних секції (рис. 9):

1. Формули – відображення відповідної математичної формули для кожного типу об'єктів;
2. Кнопки – запуск обчислень для відповідного об'єкта;
3. Параметри – поля для введення n та k (або списку кратностей для перестановок із повторенням);
4. Результат – поле виведення усіх варіантів об'єкта та їхньої кількості.

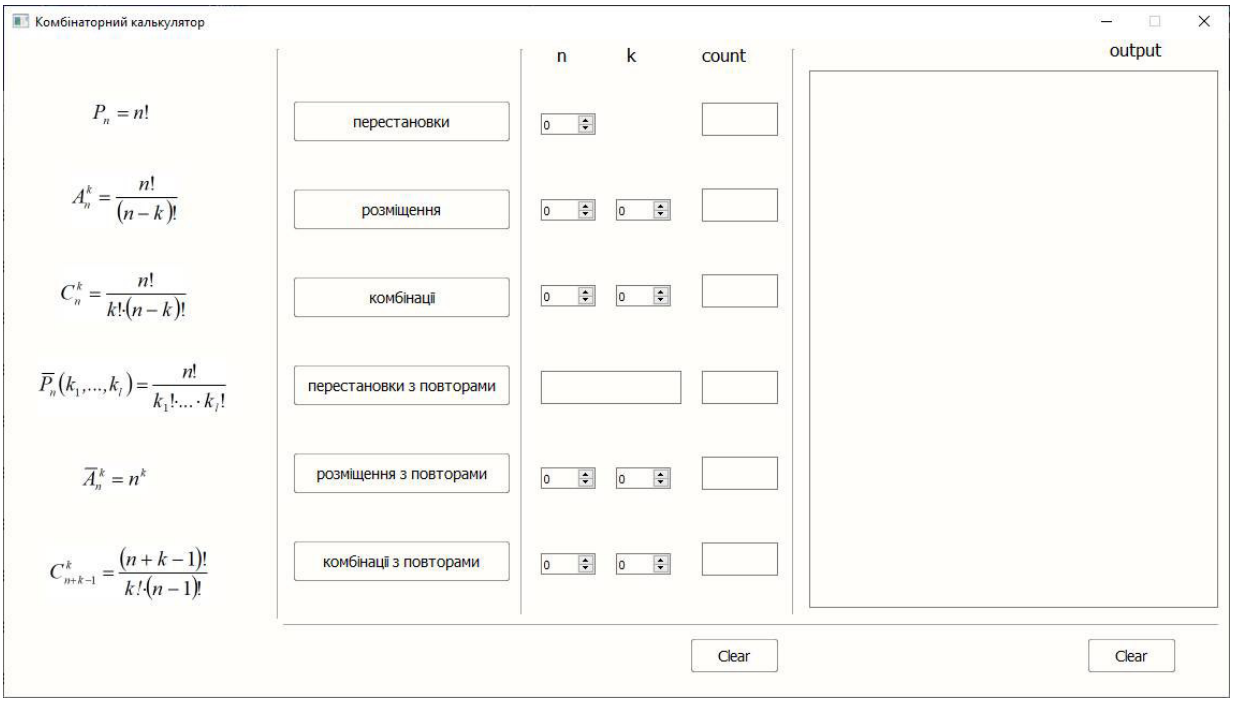


Рис. 9. Інтерфейс демонстраційної програми

Програма підтримує всі шість розглянутих типів об'єктів. Нижче наведено приклади результатів роботи.

4.4 Практичне значення

Практичне значення роботи полягає в тому, що розроблений програмний продукт може використовуватись як навчальний засіб на уроках математики та інформатики для ілюстрації понять комбінаторики, а також як калькулятор для швидкого отримання усіх варіантів комбінаторного об'єкта при невеликих значеннях параметрів.

Висновки

У статті розглянуто основні типи комбінаторних об'єктів: перестановки, розміщення та комбінації, а також їхні варіанти з повторенням. Для кожного типу наведено математичну формулу підрахунку кількості об'єктів та алгоритм їх генерації.

Реалізовано рекурсивні та ітераційні алгоритми мовою Python. Рекурсивний підхід є природним і зрозумілим, проте менш ефективним із погляду використання пам'яті при великих вхідних параметрах. Ітераційні алгоритми з генераторами є кращим вибором для практичних застосувань. Використання `yield` у Python є ефективним засобом потокової генерації великих наборів даних без надмірного навантаження на пам'ять.

Підтверджено, що задачі генерації комбінаторних об'єктів за своєю природою є NP-задачами: кількість об'єктів зростає факторіально або степенево і стає практично нездоланною вже при відносно невеликих значеннях параметрів (наприклад, 10! ≈

3.6 млн). Це пояснює, чому при розв'язанні прикладних задач оптимізації, що зводяться до перебору комбінаторних конфігурацій, застосовують евристичні та метаевристичні методи.

Розроблено програмний продукт із графічним інтерфейсом на базі бібліотеки PyQt5, який наочно демонструє роботу реалізованих алгоритмів і може застосовуватись у навчальному процесі.

Список використаної літератури

1. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. – 4th ed. – Cambridge, MA: The MIT Press, 2022. – 1312 p.
2. PyQt5 5.15.10 [Електронний ресурс]. – Режим доступу: https://pypi.org/project/PyQt5/
3. Python Programming Language [Електронний ресурс]. – Режим доступу: https://www.python.org/
4. Real Python Tutorials [Електронний ресурс]. – Режим доступу: https://realpython.com/
5. Schmidt W.M. Diophantine Approximations and Diophantine Equations. – Berlin: Springer-Verlag, 1991. – 228 p.
6. Вигоднер І.В., Білоусова Т.П., Ляхович Т.П. Теорія ймовірностей та математична статистика: навчальний посібник. – Херсон: Гельветика, 2019. – 336 с.
7. Гаврилків В.М. Формальні мови та алгоритмічні моделі. – Івано-Франківськ: Голіней, 2023. – 180 с.
8. Зайцев Є. Теорія ймовірностей і математична статистика. – Київ: Алерта, 2013. – 440 с.
9. Костарчук В. М. Теорія графів та її застосування / В. М. Костарчук. – Київ : Вища школа, 1986. – 224 с.
10. Мартинюк О.М., Попіна С.Ю. Елементи комбінаторики й класичне означення ймовірності. – Тернопіль, 2003. – 40 с.
11. Diestel R. Graph Theory / R. Diestel. – 5th ed. – Hamburg : Springer, 2017. – 428 p.
12. Підручник з Python [Електронний ресурс]. – Режим доступу: https://docs.python.org/uk/3/tutorial/index.html

References

1. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. (2022) Introduction to Algorithms (4th ed.). Cambridge, MA: The MIT Press.
2. PyQt5 5.15.10. Available at: https://pypi.org/project/PyQt5/
3. Python Programming Language. Available at: https://www.python.org/
4. Real Python Tutorials. Available at: https://realpython.com/
5. Schmidt W.M. (1991) Diophantine Approximations and Diophantine Equations. Berlin: Springer-Verlag.
6. Vyhondner I.V., Bilousova T.P., Liakhovych T.P. (2019) Probability Theory and Mathematical Statistics: a textbook. Kherson: Helvetyka. [in Ukrainian]
7. Havrylkiv V.M. (2023) Formal Languages and Algorithmic Models. Ivano-Frankivsk: Holinei. [in Ukrainian]
8. Zaitsev Ye. (2013) Probability Theory and Mathematical Statistics. Kyiv: Alerta. [in Ukrainian]
9. Kostarchuk V. M. (1986) Graph Theory and Its Applications / V. M. Kostarchuk. – Kyiv : Vyshcha Shkola. [in Ukrainian]
10. Martyniuk O.M., Popina S.Yu. (2003) Elements of Combinatorics and the Classical Definition of Probability. Ternopil. [in Ukrainian]
11. Diestel R. (2017) Graph Theory. Hamburg : Springer.
12. Python Tutorial (Ukrainian). Available at: https://docs.python.org/uk/3/tutorial/index.html

KURYLO Oleksandr Mykolaiovych,
head, teacher of Computer Science at Zolotonosha General Education School of Levels I-III No. 5

NITSAK Vladyslav Kostiantynovych,
student of Zolotonosha General Education School of Levels I-III No. 5

RESEARCH AND SOFTWARE IMPLEMENTATION OF COMBINATORIAL OBJECT GENERATION ALGORITHMS

Summary. Introduction. *Combinatorics is one of the fundamental branches of discrete mathematics concerned with counting and constructing objects that satisfy given properties. Combinatorial objects – permutations, placements, and combinations – are widely used in probability theory, statistics, cryptography, and algorithm design. The generation of all possible configurations of such objects belongs to the class of NP problems, where the number of configurations grows factorially or exponentially with input size.*

Purpose. *The aim of this work is to systematize the mathematical foundations of the main combinatorial objects, describe and compare algorithms for their generation implemented in Python using recursive and iterative approaches, and develop a software product with a graphical interface for visual demonstration.*

Results. *The paper presents mathematical formulas for counting all six types of combinatorial objects (permutations, placements, and combinations, with and without repetition). Recursive and iterative generation algorithms are described and implemented in Python. The use of generators and the yield keyword is shown to provide an efficient approach to streaming generation of large sets of combinatorial objects. The recursive approach is more readable and natural but less memory-efficient; iterative algorithms with generators are preferable for practical use. A graphical application built with PyQt5 demonstrates the algorithms' results interactively.*

Conclusion. *The generation of combinatorial objects is inherently NP in nature: the number of objects grows unmanageably fast even for moderate input values. Both recursive and iterative algorithms yield correct results; the choice between them depends on the specific requirements of readability, memory efficiency, and the size of input data. The developed software product can be used as an educational tool in mathematics and computer science classes.*

Keywords: *combinatorics, permutations, placements, combinations, recursive algorithm, Python, generator.*

Одержано редакцією 12.11.2024 р.
Прийнято до публікації 11.12.2024 р.

УДК 378:517

DOI 10.31651/2076-5886-2024-1-45-56

PACS 01.40.Fk, 02.30.Hq

БОСОВСЬКИЙ Микола Васильович,
кандидат педагогічних наук, доцент,
доцент кафедри математики та методики навчання математики, Черкаський національний університет імені Богдана Хмельницького
e-mail: bosovskyy@gmail.com
ORCID: 0000-0003-1187-5550

СЕРДЮК Зоя Олексіївна,
кандидат педагогічних наук, доцент,
завідувач кафедри математики та методики навчання математики, Черкаський національний університет імені Богдана Хмельницького
e-mail: serdyuk_z@ukr.net
ORCID: 0000-0002-9376-4346

ТРЕТЯК Микола Васильович,
кандидат педагогічних наук, доцент,
доцент кафедри математики та методики навчання математики, Черкаський

національний університет імені Богдана Хмельницького
e-mail: tretiak@gmail.com
ORCID: 0000-0001-8918-5467

ФОРМУВАННЯ МАТЕМАТИЧНИХ КОМПЕТЕНТНОСТЕЙ У СТУДЕНТІВ ШЛЯХОМ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ФІЗИЧНИХ ЗАДАЧ

У статті розглядається підхід до формування математичних компетентностей у студентів засобами математичного моделювання фізичних задач. Обґрунтовано актуальність інтеграції математичних та фізичних знань у навчальному процесі з метою розвитку аналітичного мислення, вміння застосовувати математичні методи для опису реальних явищ і розв'язання прикладних задач. Проаналізовано етапи математичного моделювання та їхній вплив на засвоєння математичних понять, розвиток творчих здібностей і підвищення мотивації до вивчення математики. Окрему увагу приділено прикладам задач, які можуть бути ефективними у процесі формування міжпредметних зв'язків і професійно значущих компетентностей. Зроблено висновок про доцільність систематичного використання математичного моделювання у підготовці фахівців технічного та природничого профілів.

Ключові слова: *математичний аналіз, диференціальні рівняння, комплексний аналіз, математичне моделювання, фізичні задачі, математичні компетентності, студенти ЗВО.*

Постановка проблеми. Підготовка сучасних фахівців різних галузей найчастіше вимагає симбіозу знань з різних областей знань. Нині вчитель математики, фізики, інформатики, фізик чи математик, інженер чи програміст повинні бути спроможними до поєднання теоретичних знань з математики та їх застосування до розвя'зування практичних задач з фізики, математики, хімії, біології тощо. Тобто, грунтовні знання здобувачів вищої освіти з класичних математичних дисциплін («Математичний аналіз», «Аналітична геометрія», «Лінійна алгебра», «Диференціальні рівняння» та ін.), опанування ними базовими м атематичними компетентностями та вдале їх застосування до вирішення різних практичних проблем, є гарантією формування грунтовних професійних навичок у таких спеціалістів. У сучасній системі вищої освіти особливої актуальності набуває формування математичної компетентності як складової професійної підготовки майбутніх фахівців. Проте традиційне викладання математики нерідко зводиться до засвоєння формул та шаблонного розв'язання задач, що не сприяє розвитку критичного мислення, умінь аналізувати реальні ситуації, будувати математичні моделі та інтерпретувати отримані результати. Фізика, як природнича наука, надає широкі можливості для інтеграції прикладних задач у математичну освіту. Розв'язування фізичних задач із використанням методів математичного моделювання дає змогу студентам не лише глибше засвоїти математичні поняття, а й розвивати міждисциплінарні зв'язки, аналітичне мислення та вміння застосовувати знання в реальних умовах.

Проте в педагогічній практиці бракує методичних підходів до цілеспрямованого використання фізичних задач як засобу формування математичних компетентностей. Також недостатньо досліджено, які саме компетентності найефективніше розвиваються через математичне моделювання, та за яких умов цей процес є найбільш результативним. Це обумовлює потребу в ґрунтовному дослідженні можливостей застосування математичного моделювання фізичних задач у процесі навчання математики з метою формування ключових математичних компетентностей студентів.

Аналіз останніх досліджень та публікацій. Основи сучасної методології математичного моделювання були започатковані у працях таких науковців, як В. Глушков, Б. Гнеденко, А. Колмогоров, В. Королюк, В. Остапенко, О. Самарський, та

інших. Важливість використання математичного моделювання під час навчання математики зазначено у роботах українських методистів (І. Акуленко, І. Богатирьова, В. Волошена, О. Гриб'юк О, Г. Катеринюк, Т. Крилова, Л. Панченко, А. Прус, Л. Соколенко, Н. Тарасенкова, М. Філімонова, Л. Філон, В. Швець) та зарубіжних науковців (W. Blum, R. Borromeo Ferri, H. Burkhardt, G. Kaiser, K. Maaß, H. Pollak [1, 2, 3, 4]) та інших.

Метою статті є дослідження процесу формування математичних компетентностей у студентів через застосування математичного моделювання фізичних задач, аналіз ефективності використання математичних інструментів та методів для розвитку критичного мислення та аналітичних навичок у навчальному процесі.

Виклад основного матеріалу. Математичне моделювання є важливим інструментом пізнання, що дозволяє студентам не лише аналізувати реальні фізичні процеси, але й розвивати гнучке математичне мислення, логіку та здатність до формалізації. У контексті формування математичних компетентностей студентів математичне моделювання виконує роль інтегратора знань з математики та фізики (хімії, біології, географії та ін.), сприяючи глибшому розумінню міжпредметних зв'язків.

Процес математичного моделювання включає кілька основних етапів [5, 6].

1. Постановка задачі – формулювання проблеми з урахуванням умов та обмежень.

2. Створення математичної моделі – перехід від опису того чи іншого явища до його математичної інтерпретації (через рівняння, функції, графіки, неформальні залежності тощо).

3. Аналіз моделі – розв'язання моделі аналітичними або числовими методами.

4. Інтерпретація результатів – переведення математичного результату у контекст, відповідний до умови задачі (фізичний, хімічний, біологічний тощо).

5. Перевірка адекватності моделі – зіставлення результатів моделювання з експериментальними або теоретичними даними.

Залучення студентів різних спеціальностей (природничо-математичних, інженерних, ІТ, та ін.) до розв'язання фізичних задач за допомогою математичного моделювання стимулює формування таких ключових математичних компетентностей, як: 1) математична грамотність – здатність інтерпретувати, формулювати та розв'язувати проблеми, що мають математичну природу; 2) уміння застосовувати математичні методи на практиці – використання рівнянь, графіків, функцій для опису реальних явищ; 3) критичне мислення – аналіз припущень, перевірка гіпотез, вибір оптимального методу розв'язання; 4) комунікативні навички – пояснення ходу моделювання, інтерпретація результатів у доступній формі.

Таким чином, математичне моделювання фізичних задач є не лише ефективним методом засвоєння математичних понять, а й потужним засобом розвитку математичних компетентностей, необхідних для професійної діяльності майбутніх фахівців. Деякі аспекти застосування математичного інструментарію до розв'язування прикладних задач з курсів математичного аналізу та оптики описано нами у роботах [7, 8].

Розглянемо задачу з класичного курсу математичного аналізу, створимо її математичну модель за вказаними вище етапами.

Задача 1 [9]. Куля входить у дошку товщиною h = 10 см зі швидкістю $\vartheta_0 = 200$ м/с, а вилітає з дошки, пробивши її, зі швидкістю $\vartheta_1 = 80$ м/с. Припускаючи, що сила опору дошки рухові кулі пропорціональна квадратові швидкості руху, знайти, протягом якого часу триває рух кулі через дошку.

1. *Постановка задачі*. Відповідно до умови задачі маємо об'єкт – це куля, масу якої позначимо через m. Цей об'єкт проходить деякий шлях, який позначимо через x за

час *t*. Спроектуємо нашу фізичну ситуацію на геометричну площину, при цьому будемо вважати, що початок декартової системи координат співпадає з початком входження кулі в дошку. Зображуємо всі сили, які діють на кулю. ($\overline{P}$ – сила тяжіння, $\overline{N}$ – сила протидії, $\overline{F_{on}}$ – сила опору) (рис.1).

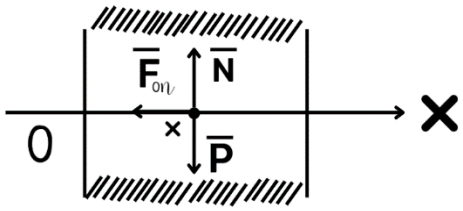


Рис.1

Таким чином, ми потлумачили умову даної задачі математичною мовою з використанням певних фізичних властивостей даного об'єкта.

2. *Створення математичної моделі.*

Згідно другого закону Ньютона $m\overline{a} = \sum_i \overline{F_i}$. в проекції на вісь *OX* маємо $m\frac{d^2x}{dt^2} = -k(\frac{dx}{dt})^2$ так, як $F_{on} = -k(\frac{dx}{dt})^2$ згідно умови задачі, або ж $\frac{d^2x}{dt^2} = -\frac{k}{m}(\frac{dx}{dt})^2$.

Таким чином, математичною моделлю даної задачі є диференціальне рівняння 2-го порядку, яке нам і треба розв'язати.

3. *Аналіз моделі*.

Для того, щоб розв'язати отримане диференціальне рівняння, зробимо у ньому наступну заміну: $\frac{dx}{dt} = \vartheta$. Підставивши в дане рівняння, отримаємо диференціальне рівняння першого порядку з відокремлюваними змінними, а саме: $\frac{d\vartheta}{dt} = -\frac{k}{m}\vartheta^2$. Відокремивши змінні, матимемо: $\frac{d\vartheta}{\vartheta^2} = -\frac{k}{m}dt$.

Інтегруємо обидві частини останнього рівняння: $\int\frac{d\vartheta}{\vartheta^2} = \int\frac{k}{m}dt$, отримаємо: $\frac{1}{\vartheta} = \frac{k}{m}t + C_1$ або $\vartheta = \frac{1}{\frac{k}{m}t+C_1}$.

Згідно з початковими умовами $\vartheta_0 = 200$ м/с в початковий момент $t = 0$, тому, підставивши їх у загальний розв'язок, ми можемо визначити C_1: $\frac{1}{200} = \frac{k}{m}\cdot 0 + C_1$, $C_1 = \frac{1}{200}$.

Звідси залежність швидкості від часу матиме вигляд: $\vartheta = \frac{1}{\frac{k}{m}t+\frac{1}{200}}$; $\vartheta = \frac{1}{1+200\frac{k}{m}t}$.

Оскільки швидкість є похідною від шляху, то замінимо її на відповідний вираз: $\vartheta = \frac{dx}{dt}$, звідки отримаємо знову диференціальне рівняння з відокремлюваними змінними, розв'яжемо його аналогічно: $\frac{dx}{dt} = \frac{200}{1+200\frac{k}{m}t}$; $dx = \frac{200dt}{1+200\frac{k}{m}t}$, $x = \frac{m}{k}\ln(1+200\frac{k}{m}t) + C_2$. Використавши ще раз початкову умову, тобто $x = 0$ при $t = 0$, отримаємо, що $C_2 = 0$.

Отже, залежність відстані від часу, яку пройде куля в дошці матиме вигляд:

$$x = \frac{m}{k}\ln(1+200\frac{k}{m}t) + C_2.$$

Підставивши $\vartheta = 80$ м/с і x = 0,1м у відповідні рівності, отримаємо систему з невідомими $\frac{k}{m}$ та t:

$$\begin{cases} 80 = \dfrac{200}{1 + 200\frac{k}{m}t} \\ 0{,}1 = \dfrac{m}{k}\ln(1 + 200\dfrac{k}{m}t) \end{cases}$$

Виразивши з першого рівняння $1 + 200\frac{k}{m}t - \frac{5}{2}$ і підставивши дане значення у друге рівняння, отримаємо, що $0{,}1 = \frac{m}{k}\ln\frac{5}{2}$, звідси $\frac{k}{m} = 10\ln\frac{5}{2}$. Підставляємо отримане значення у перше рівняння системи і знайдемо час, протягом якого куля рухається в дошці:

$$80 = \frac{200}{1+200\cdot 10\ln\frac{5}{2}t};$$

$$\frac{5}{2} - 1 + 2000\ln\frac{5}{2}t;$$

$$\frac{3}{2} - 2000\ln\frac{5}{2}t;$$

$$t = \frac{3}{2\cdot 2000\ln\frac{5}{2}} \approx 0{,}0008.$$

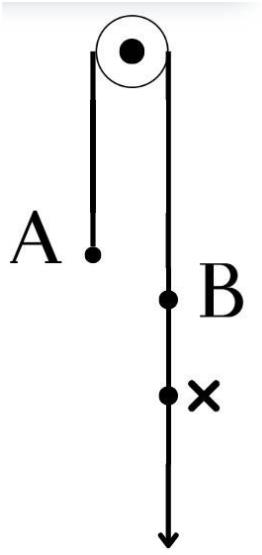


Рис. 2

4. *Інтерпретація результатів.*

Отриманий розв'язок останньої системи рівнянь і є шуканим часом, протягом якого триває рух кулі через дошку, а саме: $t \approx 0{,}0008$ с.

5. *Перевірка адекватності моделі.*

Отриманий результат є достовірним, тому і є розв'язком даної задачі.

Отже, отримали відповідь: $t \approx 0{,}0008$ с.

Дану задачу варто пропонувати студентам після вивчення курсу математичного аналізу, зокрема після вивчення теми «Інтегральне числення функції однієї змінної», та або під час вивчення курсу «Диференціальні рівняння», або ж наприкінці його в якості закріплення вивченого матеріалу.

Далі розглянемо задачу вже з класичного курсу механіки та за аналогічним принципом побудуємо її математичну модель.

Задача 2 [10]. Мотузка висить на гладенькому блоці. В початковий момент по один бік звисає 10 м мотузки, а по інший 8 м. Сила опору не враховується. Через скільки часу мотузка зісковзне з блоку?

1. *Постановка задачі*. Для того, щоб краще розібратися в умові задачі та описати її математичну модель, необхідно зробити відповідний рисунок (рис. 2). Блок з мотузкою розмістимо у систему координат так, що спрямуємо вісь *OX* вниз. Блок обираємо як точку відліку. Тоді за умовою задачі *AO* = 8 м, *BO* = 10 м. Нехай в момент часу t точка *B* при русі знаходиться на відстані x від початку координат.

2. *Створення математичної моделі.*

Відповідно до другого закону Ньютона $m\bar{a} = \sum_i \overline{F_i}$ маємо $m\bar{a} = \overline{F_1} + \overline{F_2}$, де $\overline{F_1}$ – сила тяжіння правої частини мотузки, а $\overline{F_2}$ – лівої.

Звідси

$$F_1 = m_1 \cdot g = x \cdot g \cdot \gamma,$$

$$F_2 = -m_2 \cdot g = -(18 - x)\gamma g,$$

де γ – лінійна густина мотузки , g – прискорення вільного падіння. Отже, маємо:

$$18 \cdot \gamma \cdot \frac{d^2x}{dt^2} = x \cdot g \cdot \gamma - (18 - x)\gamma g,$$

$$18\frac{d^2x}{dt^2} = (2x - 18)g,$$

$$9\frac{d^2x}{dt^2} = (x - 9)g.$$

Таким чином наша математична модель – це лінійне неоднорідне диференціальне рівняння 2-го порядку.

3. *Аналіз моделі*.

Для того, щоб розв'язати отримане диференціальне рівняння, перепишемо його в більш зручному вигляді та складемо характеристичне рівняння до нього:

$$9\frac{d^2x}{dt^2} - xg = -9g$$

$$\frac{d^2x}{dt^2} - \frac{g}{9}x = -g$$

Характеристичне рівняння: $k^2 - \frac{g}{9} = 0$ (Позначимо $\frac{g}{9} = a^2, a = \sqrt{\frac{g}{9}} \in R$), тоді отримаємо:

$$k_1 = a\, k_2 = -a,$$

$$x_1 = e^{at}, x_2 = e^{-at}.$$

Загальний розв'язок даного рівнянням матиме вигляд: $x = C_1 e^{\sqrt{\frac{g}{9}}t} + C_2 e^{-\sqrt{\frac{g}{9}}t}$.

Знайдемо константи, використовуючи початкові умови задачі, а саме: $x = 10$, $t = 0$, $\vartheta = 0$. Для цього замінимо швидкість $\vartheta = \frac{dx}{dt}$; $\vartheta = \frac{dx}{dt} = \sqrt{\frac{g}{9}} \cdot C_1 e^{\sqrt{\frac{g}{9}}t} \cdot C_2 e^{-\sqrt{\frac{g}{9}}t}$.

Отже, $\begin{cases} 10 = C_1 + C_2, \\ 0 = \sqrt{\frac{g}{9}}C_1 - \sqrt{\frac{g}{9}}C_2; \end{cases}$ $\begin{cases} 10 = C_1 + C_2, \\ 0 = C_1 - C_2; \end{cases}$ звідки $C_1 = 5$, $C_2 = 5$.

Закон зміни шляху від часу матиме вигляд: $x = 5e^{\sqrt{\frac{g}{9}}t} + 5e^{-\sqrt{\frac{g}{9}}t} = 10ch\sqrt{\frac{g}{9}}t$.

Підставимо $x = 18$ (м) та отримаємо $18 = 10ch\left(\sqrt{\frac{g}{9}}t\right)$,

звідси : $ch\left(\sqrt{\frac{g}{9}}t\right) = \frac{9}{5}$, $\sqrt{\frac{g}{9}}t = Arch\frac{9}{5}$, $t = Arch2{,}5 \cdot \sqrt{\frac{g}{t}}$, $t \approx 1{,}5$.

4. *Інтерпретація результатів.*

Отриманий розв'язок диференціального рівняння з початковими умовами і є шуканим часом, через який мотузка зісковзне з даного блоку, а саме: $t \approx 1{,}5$ с.

5. *Перевірка адекватності моделі.*

Отриманий результат є достовірним, тому і є розв'язком даної задачі.

Отже, отримали відповідь: $t \approx 1{,}5$ с.

Далі розглянемо дещо складнішу задачу. Для її розв'язання необхідні більш ґрунтовні знання як з фізики, так і з курсів математичного аналізу, диференціальних рівнянь, комплексного аналізу. Тобто насправді розв'язування даної задачі є синергію набутих здобувачами компетентностей з різних навчальних дисциплін, а моделювання задачі спрямоване на відтворення знань з різних розділів математики та фізики.

Задача 3. Коливальний контур складається з ємності *C,* індуктивності *L*, активного опору *R*. Конденсатор заряджається від постійного джерела напругою *U* в початковий момент (вимикач в положенні 1). Потім вимикач переводиться в положення 2 (рис. 3) і в замкнутому електричному колі відбуваються електромагнітні коливання (енергія електричного поля конденсатора переходить в енергію магнітного поля котушки і навпаки). При цьому частина енергії втрачається на активному опорі *R* (результуючий активний опір всіх складових) і величина напруги на конденсаторі зменшується. Знайти закон: 1) зміни напруги на конденсаторі *u*; 2) зміни струму в контурі *i*; 3) зміни заряду конденсатора *q*.

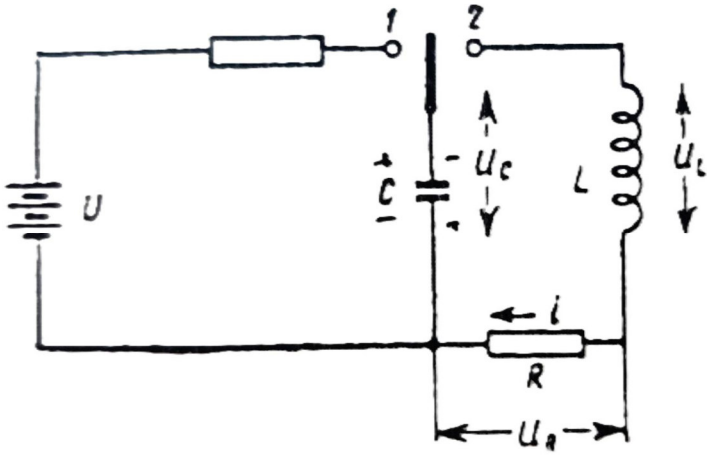


Рис. 3

1. *Постановка задачі*.

Відповідно до умови задачі, коливальний контур, зображений на рисунку 3, складається з ємності *C,* індуктивності *L*, активного опору *R*. Коли вимикач переводять з положення 1 в положення 2, то в замкнутому електричному колі відбуваються електромагнітні коливання. При цьому частина енергії втрачається на активному опорі *R* (результуючий активний опір всіх складових) і величина напруги на конденсаторі зменшується.

2. *Створення математичної моделі.*

Згідно закону Ома струм *i* в контурі визначається як частка від ділення напруги на величину опору, тобто: $i = \dfrac{U_R}{R}$ або $i = \dfrac{U_C - U_L}{R}$.

Звідси $iR = U_C - U_L$ або ж $U_C + iR - U_L = 0$, (1)

де U_C – напруга на конденсаторі, $U_Д$ – напруга на індуктивності.

Відомо, що $U_L = L\dfrac{di}{dt}$, $U_C = \dfrac{q}{c}$, $i = -\dfrac{dq}{dt}$.

Звідси $\frac{di}{dt} = -\frac{d^2q}{dt^2}$.

Тоді рівняння (1) матиме такий вигляд: $-L\frac{d^2q}{dt^2} - R\frac{dq}{dt} - \frac{q}{c} = 0$

або ж $\frac{d^2q}{dt^2} + \frac{R}{L}\frac{dq}{dt} + \frac{1}{Lc}q = 0$. (2)

Отже, ми побудували математичну модель даної задачі у вигляді диференціального рівняння 2-го порядку (2), який подає закон зміни заряду на конденсаторі. Далі розв'яжемо його.

3. *Аналіз моделі*.

Після підстановки в (2) $q = cU$ отримаємо закон зміни напруги на конденсаторі:

$c\frac{d^2U}{dt^2} + \frac{R}{L}\frac{cdU}{dt} + \frac{cU}{Lc} = 0$, звідки після виконання елементарних дій отримаємо:

$\frac{d^2U}{dt^2} + \frac{R}{L}\frac{dU}{dt} + \frac{U}{Lc} = 0$. (3)

Продиференціюємо рівняння (2) по змінній t і отримаємо:

$\frac{d}{dt}\left(\frac{d^2q}{dt^2} + \frac{R}{L}\frac{dq}{dt} + \frac{1}{Lc}q\right) = 0$, $\frac{d^3q}{dt^3} + \frac{R}{L}\frac{d^2q}{dt^2} + \frac{1}{Lc}\frac{dq}{dt} = 0$.

Враховуючи, що $-\frac{d^3q}{dt^3} = \frac{d^2i}{dt^2}$, $-\frac{d^2q}{dt^2} = \frac{di}{dt}$, $-\frac{dq}{dt} = \frac{di}{dt}$, $-\frac{dq}{dt} = i$ отримаємо

диференціальне рівняння закону зміни струму: $\frac{d^2i}{dt^2} + \frac{R}{L}\frac{di}{dt} + \frac{i}{Lc} = 0$ (4)

Розв'язуємо диференціальне рівняння (4), складаємо характеристичне рівняння та розв'язуємо його: $k^2 + \frac{R}{L}k + \frac{1}{Lc} = 0$.

$D = \sqrt{\left(\frac{R}{L}\right)^2 - 4\frac{1}{Lc}} = 2\sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{Lc}}$.

$k_{1,2} = -\frac{R}{2L} \pm \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{Lc}}$.

Позначимо $\frac{1}{Lc} = \omega^2$, а $\frac{R}{2L} = \alpha$, тоді корені характеристичного рівняння матимуть вигляд: $k_{1,2} = -\alpha \pm \sqrt{\alpha^2 - \omega^2} = -\alpha \pm j\sqrt{\omega^2 - \alpha^2}$, де через j позначено уявну одиницю в електротехніці ($j^2 = -1$).

Отже, заряд конденсатора при знайдених коренях [11] змінюється за законом:

$$q = e^{-\alpha t}\left(C_1 \cos\sqrt{\omega^2 - \alpha^2}\,t + C_2 \sin\sqrt{\omega^2 - \alpha^2}\,t\right). \quad (5)$$

Знаходимо C_1 і C_2 з початкових умов: $t = 0$, $q = Q_{\max}$, $i = 0$.

Підставивши в (5) дані значення, отримаємо, що $Q_{\max} = C_1$.

Тому $q = e^{-\alpha t}\left(Q_{\max} \cos\sqrt{\omega^2 - \alpha^2}\,t + C_2 \sin\sqrt{\omega^2 - \alpha^2}\,t\right)$.

Звідси

$i = -\frac{dq}{dt} = e^{-\alpha t}(-\alpha Q_{\max} \cos\sqrt{\omega^2 - \alpha^2}\,t - \alpha C_2 \sin\sqrt{\omega^2 - \alpha^2}\,t - Q_{\max} \sin\sqrt{\omega^2 - \alpha^2}\,t \cdot \sqrt{\omega^2 - \alpha^2} +$

$+C_2\cos\sqrt{\omega^2-\alpha^2}t\cdot\sqrt{\omega^2-\alpha^2}),$

$i=-e^{-\alpha t}\left(\left(\sqrt{\omega^2-\alpha^2}\cdot C_2-\alpha Q_{\max}\right)\cos\sqrt{\omega^2-\alpha^2}t-\left(Q_{\max}\sqrt{\omega^2-\alpha^2}+\alpha C_2\right)\sin\sqrt{\omega^2-\alpha^2}t\right).$ (6)

При $t=0$, $i=0$ отримаємо: $\sqrt{\omega^2-\alpha^2}C_2-\alpha Q_{\max}=0$, $C_2=Q_{\max}\cdot\frac{\alpha}{\sqrt{\omega^2-\alpha^2}}$.

Отримані значення підставимо в (5) та (6) та отримаємо:

$$q=Q_{\max}e^{-\alpha t}\left(\cos\sqrt{\omega^2-\alpha^2}t+\frac{\alpha}{\sqrt{\omega^2-\alpha^2}}\sin\sqrt{\omega^2-\alpha^2}t\right), \quad (7)$$

$$i=Q_{\max}\left(\left(\sqrt{\omega^2-\alpha^2}+\frac{\alpha}{\sqrt{\omega^2-\alpha^2}}\right)e^{-\alpha t}\sin\sqrt{\omega^2-\alpha^2}t\right). \quad (8)$$

Максимальне значення струму $I_{\max}=Q_{\max}\left(\sqrt{\omega^2-\alpha^2}+\frac{\alpha^2}{\sqrt{\omega^2-\alpha^2}}\right)$, тоді рівність (8) матиме вигляд: $i=I_{\max}e^{-\alpha t}\sin\sqrt{\omega^2-\alpha^2}t$. (9)

Використавши залежність $U=\frac{q}{c}$, отримаємо закон зміни напруги на ємності:

$$U=U_{\max}e^{-\alpha t}\left(\cos\sqrt{\omega^2-\alpha^2}t+\frac{\alpha^2}{\sqrt{\omega^2-\alpha^2}}\sin\sqrt{\omega^2-\alpha^2}t\right). \quad (10)$$

4. Інтерпретація результатів.

Отже, рівності (7), (9) та (10) – це і є шукані закони зміни заряду, струму та напруги. Можна зобразити ці затухаючі коливання графічно на рисунку 4.

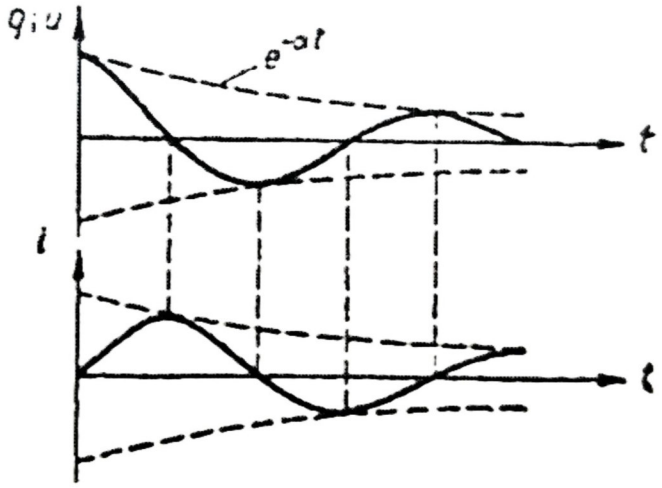


Рис. 4.

5. *Перевірка адекватності моделі.*

Отримані результати є достовірними, тому вони і є розв'язками даної задачі.

Висновки. Математичне моделювання фізичних задач є ефективним засобом розвитку математичних компетентностей студентів, оскільки сприяє формуванню здатності застосовувати математичні знання для аналізу реальних процесів, критичного мислення та самостійного розв'язання комплексних проблем.

Інтеграція фізичних задач у навчальний процес з математики дозволяє посилити міжпредметні зв'язки, актуалізувати прикладний характер математичних знань і підвищити мотивацію студентів до їх вивчення.

Застосування етапів математичного моделювання (аналіз задачі, побудова моделі, математичний розв'язок, інтерпретація результатів) формує у студентів цілісний підхід до розв'язання задач, розвиває аналітичні навички та вміння переносити знання у нові контексти.

Практичні результати впровадження методики засвідчили, що цілеспрямоване використання фізичних задач у процесі навчання математики сприяє не лише кращому засвоєнню теоретичного матеріалу, а й формуванню ключових компонентів математичної компетентності: логічного мислення, здатності до абстрагування, математичного мовлення та математичного моделювання.

Отже, математичне моделювання фізичних задач доцільно розглядати як педагогічну технологію, що забезпечує ефективне формування математичної компетентності студентів у процесі їхньої фахової підготовки.

Список використаних джерел

1. Greer, B., Verschaffel, & Mukhopadhyay. Modelling for life: Mathematics and children's experience. In W. Blum, W. Henne, & M. Niss (Eds), Applications and modelling in mathematics education. 2007. pp. 89-98. (ICMI Study 14).
2. Schukajlow S., Krawitz J., Kanefke J., Blum W., Rakoczy K. Open modelling problems: cognitive barriers and instructional prompts. Educational Studies in Mathematic. 2023.
3. Arseven A. Mathematical Modelling Approach in Mathematics EducationUniversal Journal of Educational Research 3(12), 2015. pp. 973-980.
4. Krawitz J., Kanefke J., Schukajlow S. Rakoczy K. Making realistic assumptions in mathematical modelling. In C. Fernández, S. Llinares, A. Gutiérrez, & N. Planas (Eds.). Proceedings of the 45th Conference of the International Group for the Psychology of Mathematics Education, Vol. 3, 2022. pp. 59-66.
5. Прус А. Математичне моделювання як лінза реального світу. Фізико-математична освіта, 38(4), 2023. С. 56–61. DOI 10.31110/2413-1571-2023-038-4-008.
6. Богатирьова І.М., Сердюк З.О. Методика розв'язування прикладних задач у шкільному курсі геометрії. Вісник Черкаського університету. Серія «Педагогічні науки». Черкаси: Вид. від. ЧНУ 2011. С. 19-23.
7. Тарасенкова Н. А., Сердюк З. О. Особливості викладання курсу математичного аналізу для фахівців з аналізу даних. Вісник Черкаського університету. Серія "Прикладна математика. Інформатика". 2020. № 1. Черкаси : Вид. від. ЧНУ ім. Б.Хмельницького. С. 57-73.DOI 10.31651/2076-5886-2020-1-77-86.
8. Сердюк З.О., Ткаченко А.В. Застосування математичного інструментарію до розв'язування фізичних задач. Збірник наукових праць "Актуальні питання природничо-математичної освіти". Випуск № 1 (21). Сумський державний педагогічний університет імені А.С. Макаренка, Суми, 2023. С. 77-85.
9. Дороговцев А. Я. Математичний аналіз: підручник. К.: Либідь, 1993. 320 с.
10. Диференціальні рівняння: Посібник. Розробники: В. К. Григоренко, Д.М. Лила; Черкаський національний університет імені Богдана Хмельницького. Черкаси: Вид. від. Черкаського національного університету імені Богдана Хмельницького, 2011. 232 с.
11. Кривошея С. А., Перестюк М. О., Бурим В. М. Диференціальні та інтегральні рівняння. К. Либідь, 2004. 408 с.

References:

1. Greer, B., Verschaffel, & Mukhopadhyay. (2007). Modelling for life: Mathematics and children's experience. In W. Blum, W. Henne, & M. Niss (Eds), Applications and modelling in mathematics education. [in English].
2. Schukajlow S., Krawitz J., Kanefke J., Blum W., Rakoczy K. (2023). Open modelling problems: cognitive barriers and instructional prompts. Educational Studies in Mathematic. [in English].
3. Arseven A. (2015). Mathematical Modelling Approach in Mathematics EducationUniversal Journal of Educational Research 3(12). [in English].
4. Krawitz J., Kanefke J., Schukajlow S. Rakoczy K. (2022) Making realistic assumptions in

mathematical modelling. In C. Fernández, S. Llinares, A. Gutiérrez, & N. Planas (Eds.). Proceedings of the 45th Conference of the International Group for the Psychology of Mathematics Education, Vol. 3. [in English].
5. Prus A. (2023) Mathematical modeling as a lens of the real world. Physics and mathematics education, 38(4), [in Ukrainian].
6. Bogatyreva I., Serdiuk Z. (2011) Methodology for solving applied problems in the school geometry course. Bulletin of Cherkasy University. Series "Pedagogical Sciences". Cherkasy, 19-23. [in Ukrainian].
7. Tarasenkova, N., Serdiuk, Z. (2020). Features of teaching a course of mathematical analysis for data analysis specialists. Bulletin of Cherkasy University. Series “Applied Mathematics. Informatics", 1, 57-73. Published: Mar 18, 2021. DOI 10.31651/2076-5886-2020-1-77-86. [in Ukrainian].
8. Serdiuk Z., Tkachenko A. (2023). Application of mathematical tools to solving physical problems. Collection of scientific works "Actual issues of natural and mathematical education". Issue No. 1 (21). Sumy State Pedagogical University named after A.S. Makarenko, Sumy, 77-85. [in Ukrainian].
9. Dorogovtsev, A. (1993). Mathematical analysis: [textbook] [in Ukrainian].
10. Differential equations: Manual. Developers: V. K. Hryhorenko, D. M. Lyla. (2011). Cherkasy National University named after Bohdan Khmelnytsky. – Cherkasy: Publishing house of Cherkasy National University named after Bohdan Khmelnytsky: [textbook] [in Ukrainian].
11. Kryvosheya S., Perestyuk M., Burym V. (2004). Differential and Integral Equations. K. Lybid: [textbook] [in Ukrainian].

BOSOVSKIY Mykola,
PhD (Pedagogical Sciences), Associate Professor of the Department of Mathematics and Methods of Learning of Mathematics,Cherkasy Bohdan Khmelnytsky National University
SERDIUK Zoia,
PhD (Pedagogical Sciences), Associate Professor of the Department of Mathematics and Methods of Learning of Mathematics,Cherkasy Bohdan Khmelnytsky National University
TRETIAK Mykola,
PhD (Pedagogical Sciences), Associate Professor of the Department of Mathematics and Methods of Learning of Mathematics,Cherkasy Bohdan Khmelnytsky National University

DEVELOPMENT OF STUDENTS' MATHEMATICAL COMPETENCIES THROUGH MATHEMATICAL MODELING OF PHYSICAL PROBLEMS.

Abstract. Introduction. *The article examines an approach to the development of students' mathematical competencies through mathematical modeling of physical problems. The relevance of integrating mathematical and physical knowledge in the educational process is substantiated, with the aim of fostering analytical thinking, the ability to apply mathematical methods to describe real phenomena, and to solve applied problems. The stages of mathematical modeling and their impact on the assimilation of mathematical concepts, the development of creative abilities, and the increase in motivation to study mathematics are analyzed. Particular attention is paid to examples of problems that can be effective in forming interdisciplinary connections and professionally relevant competencies. The article concludes that the systematic use of mathematical modeling is advisable in the training of specialists in technical and natural sciences.*

The purpose *of the article is to study the process of forming mathematical competencies in students through the use of mathematical modeling of physical problems, to analyze the effectiveness of using mathematical tools and methods for the development of critical thinking and analytical skills in the educational process.*

Originality. *The application of the stages of mathematical modeling (problem analysis, model construction, mathematical solution, interpretation of results) forms a holistic approach to solving problems in students, develops analytical skills and the ability to transfer knowledge to new contexts.*

The practical results of the implementation of the methodology have shown that the purposeful use of physical problems in the process of teaching mathematics contributes not only to better assimilation of theoretical material, but also to the formation of key components of mathematical competence: logical thinking, the ability to abstract, mathematical speech and mathematical

modeling.

Conclusion. *Mathematical modeling of physical problems is an effective means of developing students' mathematical competencies, as it contributes to the formation of the ability to apply mathematical knowledge to analyze real processes, critical thinking and independent solution of complex problems. Integration of physical problems into the educational process of mathematics allows to strengthen interdisciplinary connections, actualize the applied nature of mathematical knowledge and increase students' motivation to study them. Therefore, mathematical modeling of physical problems should be considered as a pedagogical technology that ensures the effective formation of students' mathematical competence in the process of their professional training.*

Keywords: *mathematical analysis, differential equations, complex analysis, mathematical modeling, physical problems, mathematical competencies, higher education students.*

Одержано редакцією 24.09.2024 р.
Прийнято до публікації 30.10.2024 р.

СЕКЦІЯ «ІНФОРМАТИКА»

УДК 004.5:004.92

DOI 10.31651/2076-5886-2024-1-57-67

PACS

ПЕДЧЕНКО Максим Анатолійович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail: pedchenko.maksym@vu.cdu.edu.ua

СЕРДЮК Олександр Анатолійович
кандидат економічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені БогданаХмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ІНТЕРАКТИВНОЇ ВІЗУАЛІЗАЦІЇ ПЕРСОНАЛЬНИХ ДАНИХ МУЗИЧНОГО СТРІМІНГУ

У статті описано архітектуру та практичну реалізацію веб-застосунку для аналізу й інтерактивної візуалізації персональних даних музичного стрімінгу платформи Spotify. Актуальність теми зумовлена стрімким зростанням обсягів даних, що генеруються користувачами стрімінгових сервісів, а також недостатністю вбудованих аналітичних засобів для їх глибокого дослідження. На підставі порівняльного аналізу існуючих рішень – сторонніх сервісів Last.fm та Stats.fm, а також вбудованих засобів Apple Music, YouTube Music і Spotify – визначено ключові обмеження наявних підходів та сформульовано вимоги до розроблюваної системи. В основу архітектурного рішення покладено принцип розподілу серверної та клієнтської частин: серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js і документоорієнтованої СУБД MongoDB, клієнтська – на основі бібліотеки React з підтримкою суворої типізації через TypeScript і централізованого керування станом засобами Redux Toolkit. Для авторизації застосовано протокол OAuth 2.0 у варіанті потоку обміну кодом авторизації, що забезпечує безпечний серверний доступ до Spotify Web API без зберігання облікових даних користувача. Повнота ретроспективних даних про прослуховування досягнута завдяки імпорту розширеної історії стрімінгу відповідно до механізму запиту персональних даних за вимогами GDPR. Отримані дані нормалізуються, фільтруються та зберігаються локально для подальшої агрегованої обробки. Візуалізацію реалізовано за допомогою бібліотеки Recharts і охоплює щомісячну динаміку прослуховувань, розподіл активності протягом доби, рейтинги виконавців, треків та альбомів з детальними індивідуальними сторінками. Наукова новизна роботи полягає у розробці комплексного підходу до інтеграції Spotify Web API з персональною аналітичною системою, що поєднує безперервний збір поточних даних та повний ретроспективний імпорт й надає користувачеві інтерактивні засоби дослідження власних музичних уподобань, недоступні в жодному з розглянутих аналогів.

***Ключові слова**: Spotify API, OAuth 2.0, MongoDB, React, TypeScript, Node.js, Redux Toolkit, Recharts, візуалізація даних, музичний стрімінг, GDPR.*

Вступ

Цифровий музичний стрімінг докорінно змінив характер взаємодії людей з

музичним контентом. Такі платформи, як Spotify, Apple Music та YouTube Music, зробили мільярди треків доступними у будь-який момент, сформувавши нові патерни споживання аудіоконтенту. Щодня сотні мільйонів користувачів генерують колосальні масиви даних: кожне прослуховування, кожен пропуск чи повтор фіксуються платформою і у своїй сукупності здатні детально описати індивідуальні музичні уподобання.

Персоналізація є ключовим елементом сучасних стрімінгових сервісів. Не лише наявність широкої музичної бібліотеки, а й здатність платформи розуміти слухача та пропонувати відповідний контент – ось що формує лояльність аудиторії. Такий рівень персоналізації досягається завдяки складним алгоритмам аналізу даних, що прогнозують майбутні уподобання на основі попередньої активності.

Попри розвиненість алгоритмічної складової, самі платформи надають вкрай обмежені засоби для того, щоб користувач міг самостійно дослідити свої дані. Щорічний звіт Spotify Wrapped охоплює лише фіксований річний інтервал і не дозволяє варіювати метрики чи часовий діапазон. Apple Music Replay функціонує аналогічно, а YouTube Music і поготів обмежується загальними агрегованими показниками без деталізації.

Сторонні сервіси аналітики – Last.fm [1] та Stats.fm [2] – частково заповнюють цю прогалину, проте мають власні суттєві обмеження. Last.fm не охоплює даних, що передують реєстрації у сервісі, а Stats.fm обмежений доступом лише до 50 останніх записів через стандартний ендпоінт Spotify API. У підсумку жоден з існуючих інструментів не надає користувачеві повної довічної ретроспективи з одночасно гнучкими та інтерактивними засобами дослідження.

Дослідження у суміжній галузі – Music Information Retrieval (MIR) – підтверджують, що аналіз музичних уподобань є ефективним інструментом для розуміння поведінки користувачів і побудови персоналізованих систем рекомендацій [5]. Проте академічні роботи здебільшого зосереджені на серверних алгоритмах, а не на розробці клієнтської аналітики, доступної кінцевому користувачеві в інтерактивному режимі.

Зазначене визначає актуальність розробки спеціалізованого веб-застосунку, що поєднує: повний імпорт ретроспективних даних прослуховування, їх збереження на власному сервері та детальну інтерактивну візуалізацію з гнучким вибором часових діапазонів.

Мета статті – описати методи проєктування та реалізації веб-застосунку для збору, зберігання й інтерактивної візуалізації персональних даних стрімінгу музики у сервісі Spotify, а також обґрунтувати обрані архітектурні рішення і стек технологій з позиції їх ефективності для вирішення поставленої задачі.

Виклад основного матеріалу

1. Огляд існуючих рішень та постановка задачі

Ринок аналітичних сервісів для стрімінгових платформ представлений двома категоріями: вбудованими засобами самих платформ і сторонніми застосунками-надбудовами, що використовують відкриті API.

Вбудовані засоби платформ. Apple Music надає функцію «Replay», що формує підсумковий перелік найпрослуховуваніших треків за поточний рік. Функціонал залишається обмеженим і не передбачає деталізованої аналітики чи користувацького налаштування параметрів. YouTube Music пропонує базову статистику уподобань і популярних треків, проте інтерактивне дослідження даних не підтримується. Spotify формує детальний щорічний звіт «Spotify Wrapped» і підтримує персоналізовані

плейлисти «Discover Weekly», однак у режимі реального часу не надає засобів для самостійного дослідження власної статистики.

Сторонні рішення. Last.fm [1] є піонерським сервісом трекінгу та аналітики прослуховувань, що фіксує активність на різних платформах через механізм «скроблінгу» і зводить дані у єдиний профіль. Основний недолік – відсутність ретроспективних даних до початку використання сервісу та залежність повноти статистики від безперебійної роботи «скроблінгу». Stats.fm [2] спрямований безпосередньо на користувачів Spotify і забезпечує докладну аналітику та візуалізацію: найпрослуховуваніші треки, виконавці, жанрові уподобання. Суттєве обмеження платформи Spotify API – доступ лише до 50 останніх записів через ендпоінт recently-played – звужує повноту даних у безкоштовному варіанті використання.

Порівняльний аналіз виявив такі системні вади розглянутих інструментів:

- неповнота ретроспективних даних – жодне рішення не охоплює повної довічної історії прослуховувань без додаткових умов;
- статичність представлення – відсутність засобів для інтерактивного дослідження та гнучкого налаштування параметрів відображення;
- обмеженість часових діапазонів – здебільшого доступні лише фіксовані інтервали (тиждень, місяць, рік) без можливості вибору довільного проміжку;
- брак аналізу внутрішньодобових патернів – жоден з розглянутих сервісів не надає деталізованого аналізу активності у розрізі годин доби.

Таким чином, задача полягає у розробці застосунку, що усуває зазначені обмеження: забезпечує повний імпорт ретроспективних даних, їх збереження та обробку на власному сервері, а також надає інтерактивні та гнучко налаштовувані засоби візуалізації.

2. Архітектура системи та стек технологій

2.1 Загальна архітектура

Розроблений застосунок побудовано за принципом клієнт-серверної архітектури. Серверна частина реалізує RESTful API [7], взаємодіє зі Spotify Web API [4] та відповідає за збереження, нормалізацію й агрегацію даних прослуховування. Клієнтська частина є SPA (Single Page Application), що відображає інтерактивні дашборди з персональною статистикою. Розподіл обов'язків між рівнями забезпечує незалежне масштабування кожного компонента та дозволяє реалізовувати складні агрегаційні запити безпосередньо на серверному рівні, а не перекладати цей обчислювальний тягар на браузер клієнта.

2.2 Серверна частина: Node.js, Express.js і MongoDB

Node.js і Express.js. Серверна частина реалізована на платформі Node.js [6] – середовищі виконання JavaScript на стороні сервера з подієво-орієнтованою неблокуючою архітектурою введення-виведення. Це рішення є обґрунтованим для застосунків з інтенсивним обміном даними, де конкурентна обробка численних запитів є критичною вимогою. Фреймворк Express.js надає зручне визначення маршрутів і ланцюжок middleware-функцій, що забезпечують обробку HTTP-запитів на різних рівнях: валідацію вхідних даних (бібліотека Zod), обробку завантажуваних файлів (Multer), аутентифікацію (перевірка JWT-токенів), логування запитів (Morgan). Модульна структура маршрутизатора (`/spotify`, `/artist`, `/album`, `/track`, `/oauth`) полегшує підтримку та розширення кодової бази.

MongoDB і Mongoose. Для зберігання даних обрано документоорієнтовану СУБД MongoDB [3], що надає гнучку схему документів і ефективно опрацьовує різнорідні дані великого обсягу. Схема бази даних включає шість основних колекцій (рис. 1):

`users` (профілі користувачів та токени авторизації), `infos` (записи про прослуховування – часова мітка, ідентифікатор треку, статус фільтрації), `tracks`, `artists`, `albums` (метадані музичного контенту, отримані від Spotify), `importstates` (журнал імпортів розширеної історії). Для роботи з MongoDB у Node.js застосовано бібліотеку Mongoose [8] – ODM-рішення (Object Data Modelling), що реалізує схемо-орієнтоване моделювання даних з вбудованою валідацією та механізмом віртуальних полів (virtuals) для реалізації зв'язків між колекціями без зберігання надлишкових посилань.

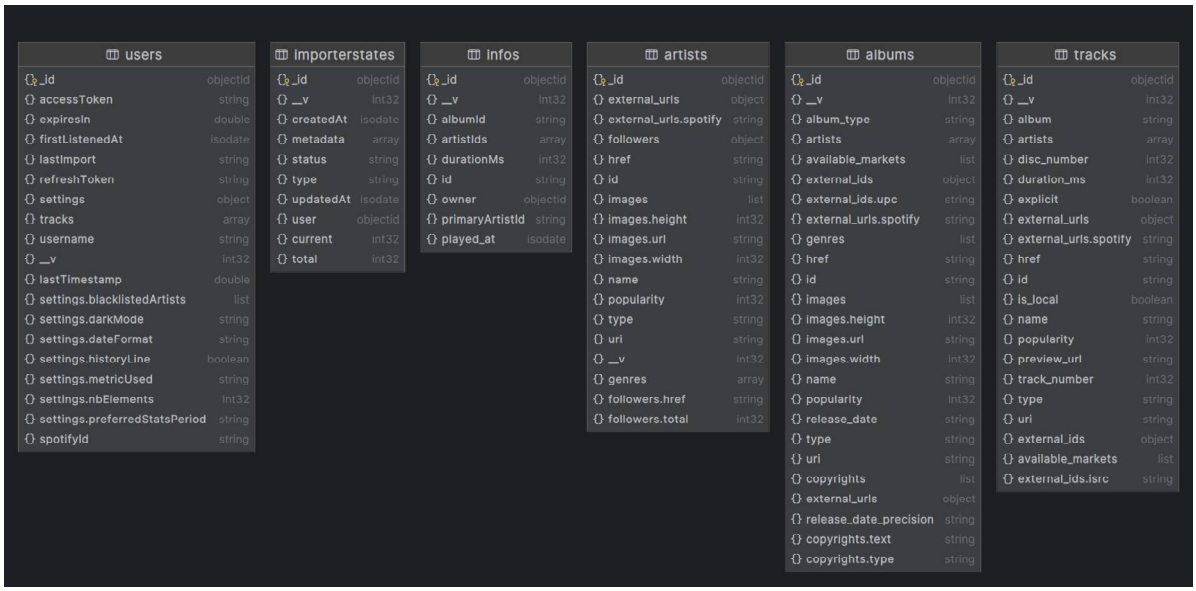


Рис. 1. Схема колекцій бази даних MongoDB

Особливу роль в архітектурі відіграє агрегаційний пайплайн MongoDB: операції `$lookup`, `$match`, `$unwind`, `$group` виконуються безпосередньо на сервері бази даних, що дозволяє ефективно обчислювати показники (наприклад, загальний час прослуховування виконавця за заданий період або розподіл активності по годинах доби) без перевантаження серверного процесу.

Для забезпечення ізоляції та відтворюваності середовища бази даних використано контейнеризацію засобами Docker із Docker Compose. Це гарантує узгодженість конфігурації на різних машинах та спрощує розгортання і міграцію.

2.3 Клієнтська частина: React, TypeScript, Redux Toolkit і Recharts

React і TypeScript. Клієнтська частина реалізована за допомогою бібліотеки React [8] у поєднанні з TypeScript [9] – суворо типізованим суперсетом JavaScript. Така комбінація підвищує надійність коду, зменшує кількість помилок часу виконання та поліпшує підтримуваність великої кодової бази. Компонентна архітектура React дозволяє будувати інтерфейс як ієрархію перевикористовуваних самодостатніх елементів зі своєю логікою та станом. JSX-розмітка компілюється у TypeScript (файли з розширенням .tsx), що дозволяє перевіряти типи пропсів компонентів на етапі компіляції.

Redux Toolkit. Для керування глобальним станом застосунку використано Redux Toolkit [9] – офіційно рекомендований спосіб написання Redux-логіки, що суттєво скорочує шаблонний код. Централізований сховок (store) охоплює зрізи (slices) для даних профілю користувача, системних повідомлень і стану імпорту. Утиліта createAsyncThunk забезпечує обробку асинхронних дій у трьох станах: pending, fulfilled,

`rejected` – що дозволяє відображати стан завантаження та помилки безпосередньо у компонентах без дублювання логіки.

Material-UI. Компонентна бібліотека Material-UI (MUI) [10] застосовується для побудови естетичного і функціонального інтерфейсу: таблиць, кнопок, діалогових вікон, навігаційних елементів, компонентів вибору дат. Система тем MUI забезпечує узгодженість візуального оформлення у всьому застосунку.

Recharts. Бібліотека Recharts [11] слугує основою для всіх інтерактивних візуалізацій. Вона нативно інтегрована з React та надає широкий набір готових компонентів: `LineChart`, `BarChart`, `AreaChart` (потоковий графік). Для забезпечення єдиного стилю й уникнення дублювання коду над стандартними компонентами Recharts побудовано три кастомні абстракції – лінійний графік, стовпчаста діаграма та складена стовпчаста діаграма, – які централізовано задають CSS-стилі, кастомні підказки (tooltips) та налаштування осей.

2.4 Авторизація через OAuth 2.0

Інтеграція зі Spotify Web API потребує аутентифікації користувача. Стандарт OAuth 2.0 [5] визначає кілька варіантів авторизаційних потоків: код авторизації, код авторизації з PKCE-розширенням, облікові дані клієнта та неявний потік. Для розробленого застосунку обрано Authorization Code Flow (рис. 2): він виконується на стороні сервера та повертає поряд із токеном доступу також токен оновлення (`refresh token`), що дає змогу підтримувати сеанс без повторного введення облікових даних.

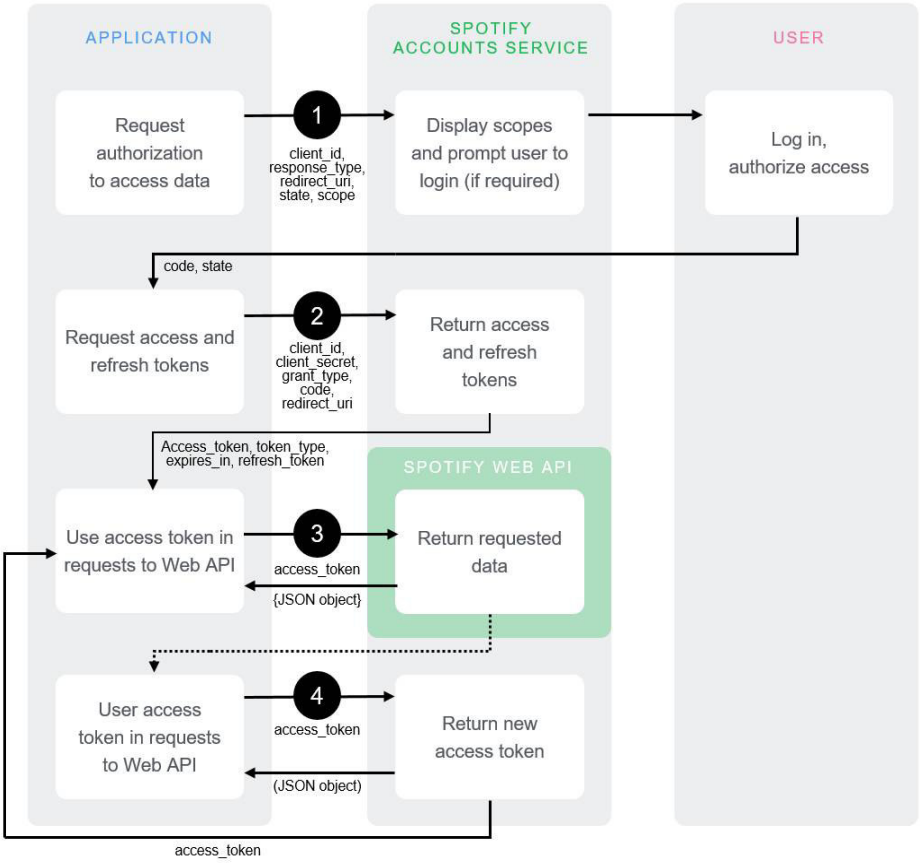


Рис. 2. Діаграма потоку авторизації OAuth 2.0

Етапи авторизації виконуються у наступній послідовності:

1. Застосунок перенаправляє користувача до сервісу Spotify Accounts з параметрами запиту дозволу.
2. Після підтвердження переліку запитуваних прав Spotify повертає одноразовий код авторизації на вказаний callback URL.
3. Сервер обмінює цей код на пару токенів (access token / refresh token), надсилаючи POST-запит безпосередньо зі свого боку – без участі клієнтського браузера.
4. Токен доступу використовується для підписаних запитів до Spotify Web API; після закінчення терміну дії він оновлюється автоматично без участі користувача.

Для захисту сеансу в межах власного застосунку отриманий токен доступу Spotify конвертується на стороні сервера у власний JWT-токен. Це мінімізує ризики витоку: навіть якщо JWT-токен буде перехоплено, зловмисник отримає доступ лише до функцій поточного застосунку, але не до облікового запису Spotify.

3. Методи отримання та обробки даних

3.1 Поточна історія прослуховувань

Для безперервного відстеження активності сервер регулярно – кожні 5 хвилин – звертається до ендпоінту Spotify Web API `api.spotify.com/v1/me/player/recently-played`. Ендпоінт підтримує параметри `after` та `before` (Unix timestamp у мілісекундах), що дозволяє щоразу отримувати лише нові записи, уникаючи дублювання. Стандартне обмеження ендпоінту – дані лише за останні 24 години – компенсується частим опитуванням.

Отримані записи про прослуховування збагачуються метаданими (виконавець, альбом, жанр, тривалість, показник популярності) через пакетний ендпоінт `api.spotify.com/v1/tracks`, що дозволяє запитувати відомості про декілька треків за один виклик API. Це суттєво знижує загальну кількість запитів та ризик перевищення ліміту.

Spotify обмежує частоту запитів на основі ковзного вікна у 30 секунд. У разі отримання статусу відповіді 429 Too Many Requests застосунок автоматично очікує 60 секунд перед повторною спробою, що забезпечує стале безперебійне опитування.

3.2 Розширена ретроспективна історія (GDPR-імпорт)

Регламент GDPR, зокрема стаття 15 [12], надає суб'єктам персональних даних право на доступ до інформації, яку контролер зберігає про них. Spotify, дотримуючись цих вимог, надає у налаштуваннях приватності можливість запитати розширену (довічну) історію стрімінгу – докладний журнал всієї активності акаунту з моменту його реєстрації.

Після обробки запиту (до 30 днів) користувач отримує ZIP-архів з JSON-файлами. Кожен об'єкт у файлах містить численні поля, але для цілей застосунку використовуються лише три: `ts` (часова мітка UTC), `ms_played` (тривалість відтворення у мілісекундах) та `spotify_track_uri` (ідентифікатор треку). Решта полів відкидається. Записи, в яких трек відтворювався менше 30 секунд, відфільтровуються як несуттєві – адже вони відповідають пропуску треку, а не його прослуховуванню.

Після нормалізації та фільтрації дані імпортуються до колекції `infos`, а стан імпорту (прогрес, кількість записів) фіксується у колекції `importstates`. Цей підхід дозволяє отримати повну ретроспективну картину музичних уподобань без залежності від моменту реєстрації у сторонніх сервісах.

4. Результати: функціональні можливості застосунку

Реалізований застосунок надає користувачеві розгалужену систему аналітичних

дашбордів, структурованих за тематичними розділами. Усі розділи підтримують гнучкий вибір часового вікна аналізу: фіксовані варіанти (останній день, тиждень, місяць, рік, весь час) і довільний діапазон через компонент вибору дат. Усі часові мітки зберігаються у форматі UTC, що забезпечує коректність відображення незалежно від часового поясу користувача.

4.1 Головна сторінка

Головна сторінка (рис. 3) формує загальний огляд музичної активності за обраний період. Зведені картки відображають кількість прослуханих треків, сумарний час прослуховування у хвилинах і кількість унікальних виконавців з порівнянням із відповідним попереднім періодом. Лінійний графік відображає часову динаміку прослуховувань, стовпчаста діаграма – розподіл активності по 24 годинах доби. Окремо виділено «найкращого виконавця» та «найкращий трек» з відповідними показниками.

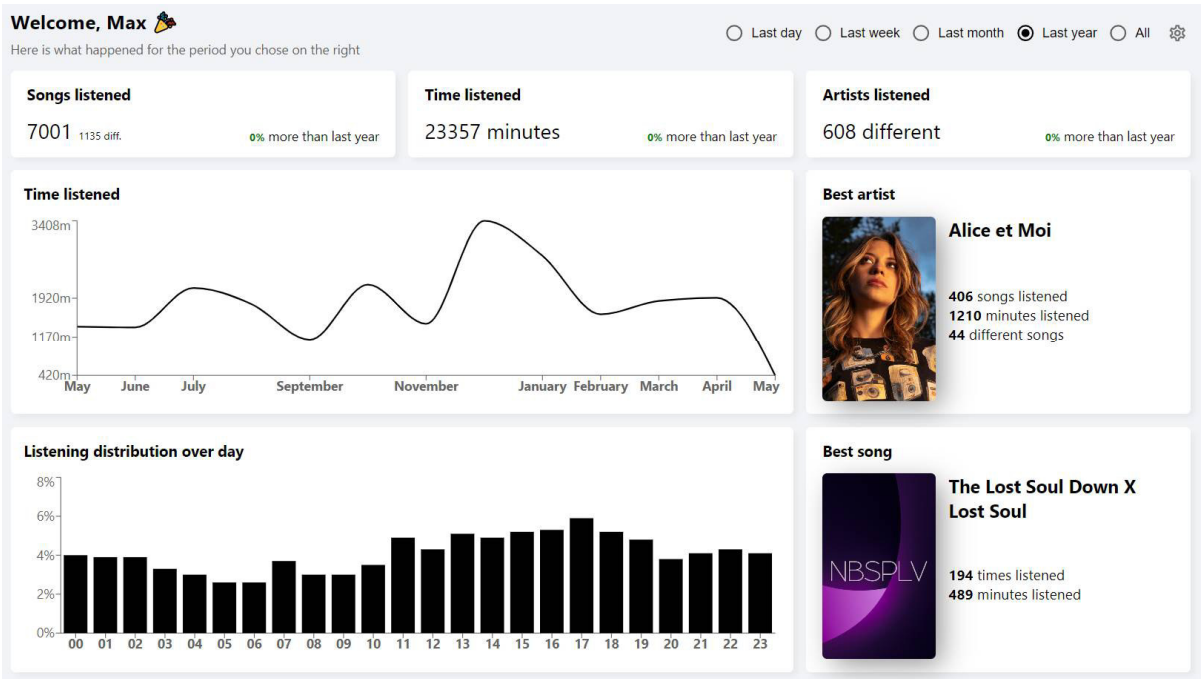


Рис. 3. Головна сторінка застосунку

4.2 Розширена статистика

Розділ «Всі статистики» включає взаємопов’язані візуалізації. Стовпчаста діаграма «Найкращі виконавці» відображає топ-виконавців з їхніми фотографіями під стовпцями для зручної ідентифікації. Потоковий графік «Розподіл прослуховування за виконавцями» ілюструє зміну часток виконавців у загальному часі прослуховування протягом року, виявляючи зсуви у музичних уподобаннях. Стовпчаста діаграма «Розподіл по годинах доби» показує відсотковий розподіл активності для виявлення пікових годин слухання. Складена стовпчаста діаграма «Найкращі виконавці для кожної години» деталізує топ-виконавців (треків або альбомів – перемикається за вибором користувача) для кожної з 24 годин.

Три додаткові лінійні графіки у щомісячному розрізі розкривають характеристики прослуховуваного контенту: середня дата виходу альбому (відображає схильність до нових релізів чи ретро), середня кількість фічерингів на трек (інтерес до колаборацій), середня популярність треків (нішові чи масові вподобання).

4.3 Рейтинги та детальна статистика

Розділ «Топи» (рис. 4) містить таблиці найпрослуховуваніших треків, виконавців і альбомів із детальними метриками: кількість відтворень (абсолютне значення та відсоток від загального), сумарний час прослуховування, жанрова приналежність (для виконавців). При переході на сторінку конкретного елемента відображається поглиблена статистика: перша та остання дата прослуховування, два місяці з найбільшою активністю, список останніх відтворень, розподіл прослуховувань по годинах доби.

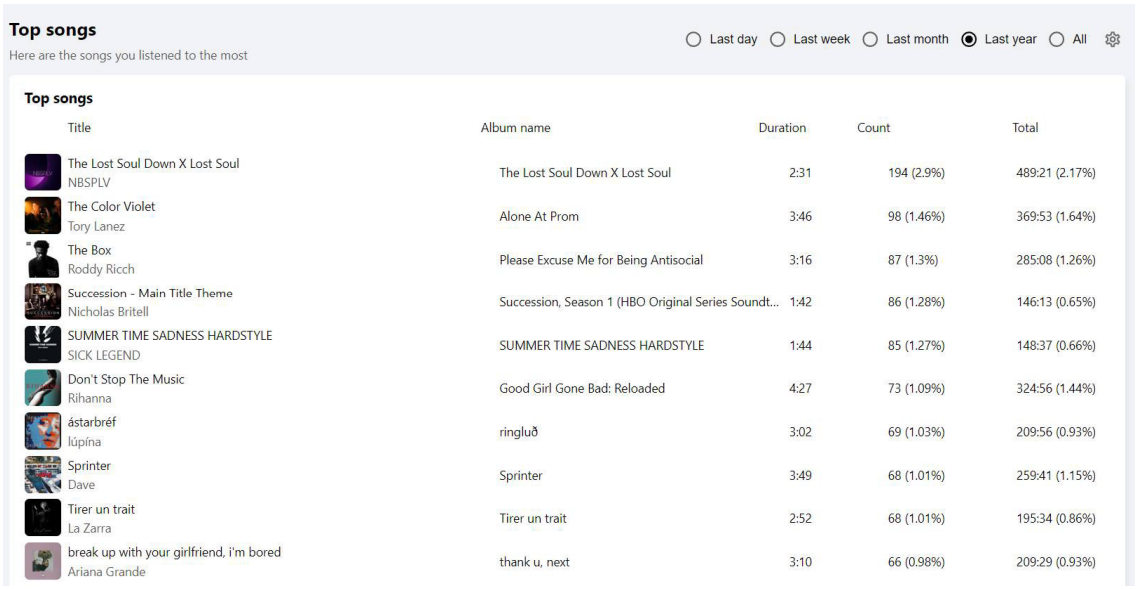


Рис. 4. Сторінка рейтингу треків

5. Аналіз результатів та наукова новизна

5.1 Порівняльний аналіз

Розроблений застосунок суттєво розширює функціональність у порівнянні з аналогами. У таблиці 1 наведено зіставлення ключових характеристик.

Найбільш принциповою відмінністю є підтримка повного GDPR-імпорту та аналізу розподілу активності за годинами доби – функцій, відсутніх у будь-якому з розглянутих аналогів.

5.2 Наукова новизна та внесок

Наукова новизна роботи включає такі складники.

По-перше, запропоновано та реалізовано методологію повного імпорту ретроспективних даних стрімінгу через механізм GDPR-запиту з подальшою нормалізацією (зниження регістру, уніфікація формату часових міток), фільтрацією (відсів відтворень коротше 30 секунд) і збереженням у власній базі даних. Такий підхід принципово відрізняється від стандартного використання Spotify API, обмеженого останніми 50 записами.

По-друге, реалізовано гібридну стратегію збору даних: регулярне опитування поточного ендпоінту (кожні 5 хвилин) забезпечує актуальність нових даних, тоді як GDPR-імпорт заповнює ретроспективний контекст від початку існування акаунту. Ця комбінація дозволяє підтримувати безперервну та повну часову шкалу прослуховувань.

По-третє, розроблено архітектурне рішення персоналізованої аналітики, у якому вся обробка даних виконується на стороні власного сервера. Це унезалежнює

аналітичні можливості від поточних обмежень Spotify API та дозволяє реалізовувати агрегаційні запити довільної складності.

По-четверте, впроваджено аналіз часових патернів у розрізі годин доби – що розкриває нові виміри музичної поведінки (пікові години слухання, прив'язаність певних виконавців до певного часу доби), недоступні в стандартних аналітичних сервісах.

Таблиця 1

Порівняння функціональних можливостей систем аналізу даних Spotify

Характеристика	Last.fm	Stats.fm	Spotify Wrapped	Розроблений застосунок
Повна ретроспективна історія	Ні	Частково	Лише за рік	Так
Інтерактивна візуалізація	Часткова	Так	Ні	Так
Довільний часовий діапазон	Ні	Частково	Ні	Так
Аналіз за годинами доби	Ні	Ні	Ні	Так
Детальна сторінка виконавця	Часткова	Так	Ні	Так
Аналіз характеристик контенту	Ні	Ні	Ні	Так
Підтримка GDPR-імпорту	Ні	Ні	–	Так

5.3 Обмеження та перспективи розвитку

Серед поточних обмежень застосунку слід відзначити: ліміт у 25 користувачів без розширення квоти розробника Spotify; відсутність підтримки декількох стрімінгових платформ одночасно; аналітика доступна лише для музики, прослуханої через Spotify.

Перспективними напрямами подальшого розвитку є: (а) застосування методів машинного навчання для автоматичного виявлення прихованих закономірностей у музичних уподобаннях та побудова прогностичних моделей активності; (б) реалізація порівняльного аналізу між двома профілями користувачів; (в) інтеграція з Last.fm API для збагачення даних про жанри та суміжних виконавців; (г) розширення квоти розробника для підтримки ширшої аудиторії.

Висновки

У статті описано процес проєктування та реалізації веб-застосунку для аналізу й

інтерактивної візуалізації персональних даних музичного стрімінгу Spotify. Проведений порівняльний аналіз існуючих рішень показав, що жодне з них не надає одночасно повної ретроспективної історії, гнучкого вибору часового діапазону та глибокої інтерактивної аналітики.

Для усунення виявлених обмежень розроблено застосунок на основі клієнт-серверної архітектури: серверна частина на Node.js/Express.js і MongoDB забезпечує збереження та ефективну агрегацію даних, клієнтська частина на React/TypeScript/Redux Toolkit і Recharts надає інтерактивні дашборди з багатовимірною аналітикою. Авторизацію реалізовано через OAuth 2.0 Authorization Code Flow, що забезпечує безпечний серверний доступ без зберігання облікових даних у клієнтській частині. Унікальною функціональною особливістю є підтримка повного імпорту ретроспективних даних через GDPR-запит у поєднанні з регулярним опитуванням поточного ендпоінту.

Реалізований застосунок надає користувачам принципово нові можливості для дослідження музичних уподобань: розподіл активності за годинами доби та місяцями, динаміка зміни улюблених виконавців протягом року, характеристики прослуховуваного контенту (середня популярність, вік релізів, кількість фічерингів), детальна статистика окремих виконавців, треків і альбомів. Отримані результати демонструють ефективність обраної архітектури та відкривають перспективи для подальшого розширення аналітичних можливостей – зокрема в напрямку застосування методів машинного навчання для виявлення прихованих закономірностей у музичній поведінці користувачів.

Список використаної літератури:

1. Last.fm | Play music, find songs, and discover artists [Електронний ресурс]. – Режим доступу: https://last.fm
2. Stats.fm (Formerly Spotistats for Spotify) [Електронний ресурс]. – Режим доступу: https://stats.fm
3. MongoDB Documentation [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs
4. Spotify Web API | Spotify for Developers [Електронний ресурс]. – Режим доступу: https://developer.spotify.com/documentation/web-api
5. Schedl M., Gómez E., Urbano J. Music Information Retrieval: Recent Developments and Applications // Foundations and Trends in Information Retrieval. – 2014. – Vol. 8, No. 2-3. – P. 127–261.
6. Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/docs/latest/api
7. Lim G. Beginning Node.js, Express & MongoDB Development. – Independently Published, 2019.
8. React Documentation [Електронний ресурс]. – Режим доступу: https://react.dev/learn
9. Getting Started | Redux Toolkit [Електронний ресурс]. – Режим доступу: https://redux-toolkit.js.org/introduction/getting-started
10. Overview – Material UI [Електронний ресурс]. – Режим доступу: https://mui.com/material-ui/getting-started
11. Recharts. Redefined chart library built with React and D3 [Електронний ресурс]. – Режим доступу: https://recharts.org
12. Article 15 GDPR. Right of access by the data subject | GDPR-Text.com [Електронний ресурс]. – Режим доступу: https://gdpr-text.com/read/article-15

References:

1. Last.fm | Play music, find songs, and discover artists. [Electronic resource]. – Access: https://last.fm
2. Stats.fm (Formerly Spotistats for Spotify). [Electronic resource]. – Access: https://stats.fm
3. MongoDB Documentation. [Electronic resource]. – Access: https://www.mongodb.com/docs
4. Spotify Web API | Spotify for Developers. [Electronic resource]. – Access: https://developer.spotify.com/documentation/web-api
5. Schedl M., Gómez E., Urbano J. Music Information Retrieval: Recent Developments and Applications. Foundations and Trends in Information Retrieval. 2014. Vol. 8, No. 2-3. P. 127–261.
6. Node.js Documentation. [Electronic resource]. – Access: https://nodejs.org/docs/latest/api
7. Lim G. Beginning Node.js, Express & MongoDB Development. Independently Published. 2019.
8. React Documentation. [Electronic resource]. – Access: https://react.dev/learn

9. Getting Started | Redux Toolkit. [Electronic resource]. – Access: https://redux-toolkit.js.org/introduction/getting-started
10. Overview – Material UI. [Electronic resource]. – Access: https://mui.com/material-ui/getting-started
11. Recharts. Redefined chart library built with React and D3. [Electronic resource]. – Access: https://recharts.org
12. Article 15 GDPR. Right of access by the data subject. [Electronic resource]. – Access: https://gdpr-text.com/read/article-15

PEDCHENKO Maksym,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

ARCHITECTURE AND IMPLEMENTATION OF A WEB APPLICATION FOR INTERACTIVE VISUALISATION OF PERSONAL MUSIC STREAMING DATA

Summary. Introduction. *The rapid growth of music streaming services has led to an accumulation of large volumes of personal data about users' listening habits. Despite the availability of basic analytics features within platforms such as Spotify, Apple Music, and YouTube Music, these tools are largely static and do not support interactive exploration or flexible time-range selection. Third-party services such as Last.fm and Stats.fm partially address this gap, yet they are limited by incomplete retrospective data coverage and constrained analytical capabilities.*

Purpose. *The purpose of this article is to describe the design principles and implementation details of a web application for collecting, storing, and interactively visualising personal Spotify music streaming data, and to justify the chosen technology stack and architectural decisions from the perspective of their effectiveness.*

Results. *A full-stack web application was developed using a client-server architecture. The backend is implemented on Node.js and Express.js with a MongoDB database managed via Mongoose, enabling scalable storage and complex server-side aggregation. User authentication is handled through the OAuth 2.0 Authorization Code Flow, which provides secure server-side access to the Spotify Web API along with refresh tokens for seamless session maintenance. A hybrid data collection strategy was implemented: regular polling of the recently-played endpoint every five minutes ensures up-to-date data, while complete retrospective history is obtained via GDPR data requests from Spotify – a combination unavailable in any existing analogue. The frontend is built with React and TypeScript, with Redux Toolkit for state management, Material-UI for interface components, and Recharts for interactive chart rendering. Three custom chart abstractions (line chart, bar chart, stacked bar chart) provide consistent styling across all dashboards. The application delivers multi-faceted analytical views: listening activity over time, hourly distribution charts, dynamic rankings of artists/tracks/albums with detailed individual statistics pages, stream graphs of artist distribution, and charts of content characteristics (average album release date, number of features per track, song popularity). Flexible date range selection – including fully custom intervals – is supported across all views.*

Conclusion. *The developed application addresses the key limitations of existing solutions by providing full retrospective listening history via GDPR import, flexible time-range analysis, and hourly-level activity breakdown. The proposed hybrid data collection methodology and the architectural approach of server-side aggregation represent contributions to the field of personalised music analytics. The results open perspectives for further development, including the integration of machine learning methods for automatic discovery of hidden patterns in users' musical behaviour.*

Keywords: *Spotify API, OAuth 2.0, MongoDB, React, TypeScript, Node.js, Redux Toolkit, Recharts, data visualisation, music streaming, GDPR.*

Одержано редакцією 09.09.2024 р.
Прийнято до публікації 30.10.2024 р.

УДК 004.852:004.8

DOI 10.31651/2076-5886-2024-1-68-79

PACS 02.60.-x, 07.05.Mh

ПІСКУН Олександр Варфоломійович
кандидат технічних наук, доцент,
завідувач кафедри прикладної математики та інформатики, Черкаський національний університет ім. Б. Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

КРАСНОШЛИК Наталія Олександрівна
кандидат технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: krasnoshlyk@vu.cdu.edu.ua
ORCID 0000-0003-4661-6997

СВІРЕНКО Данило Євгенійович
розробник ПЗ, м. Черкаси

ЗАСТОСУВАННЯ ДЕРЕВ РІШЕНЬ З НЕЙРОННОЮ ПІДТРИМКОЮ У ЗАДАЧАХ ШТУЧНОГО ІНТЕЛЕКТУ

У статті досліджено гібридні моделі машинного навчання, що поєднують дерева рішень із нейронними мережами з метою одночасного досягнення високої точності прогнозування та інтерпретованості рішень. Проаналізовано сучасні архітектури таких моделей: Neural Decision Trees (NDT), Differentiable Decision Trees, Neural Oblivious Decision Trees (NODE), TabNet та Neural-Backed Decision Trees (NBDT). Практична реалізація включає побудову гібридної моделі на датасетах Iris та нобелівських лауреатів з використанням SHAP-аналізу для інтерпретації отриманих результатів, що підтверджує ефективність та практичну застосовність описаного підходу.

Ключові слова: *дерева рішень, нейронні мережі, гібридні моделі, м'які дерева рішень, NBDT, SHAP, машинне навчання, інтерпретованість.*

Вступ

Задачі класифікації та регресії є центральними у сучасному машинному навчанні. Два найпоширеніших підходи до їх розв'язання – дерева рішень і нейронні мережі – мають принципово різні властивості та взаємодоповнювальні обмеження.

Дерева рішень є прозорими та легко інтерпретованими моделями: кожен крок логічно обґрунтований і наочно представлений, а структуру дерева можна буквально «прочитати» як схему [1]. Однак у реальних задачах, де ознаки взаємопов'язані, дані містять шум або складні залежності, дерево рішень може бути недостатньо потужним [2]. Нейронні мережі, навпаки, демонструють широкі апроксимаційні можливості та здатні виявляти приховані взаємозв'язки, які людина не може легко помітити або описати [3, 4]. Однак низька інтерпретованість нейронних мереж суттєво обмежує їх застосування у критично важливих областях.

Ця фундаментальна суперечність стимулює розвиток гібридних підходів, що поєднують переваги обох методів. Ансамблі дерев (Random Forest, Gradient Boosting), нейросимвольні методи, м'які диференційовані дерева рішень та бібліотека NBDT – все

це різні варіанти вирішення однієї проблеми балансу між точністю та пояснюваністю [5, 6].

Метою даного дослідження є дослідження теоретичних основ і сучасних архітектур гібридних моделей, що поєднують дерева рішень із нейронними мережами, практична реалізація таких моделей на задачах класифікації та оцінка інтерпретованості отриманих рішень засобами SHAP-аналізу.

Виклад основного матеріалу

1. Дерева рішень і нейронні мережі у машинному навчанні

1.1. Класичні дерева рішень

Дерево рішень – це модель машинного навчання, що використовується для розв'язання задач класифікації або регресії і працює за принципом послідовного прийняття рішень. На кожному етапі модель ставить запитання про певну ознаку об'єкта: «Чи значення ознаки більше порогу?», «Чи виконується умова?». Залежно від відповіді модель переходить до наступного запитання, доки не дійде до остаточного рішення – прогнозу [2].

Класичні алгоритми побудови дерев становлять основу багатьох алгоритмів машинного навчання. Алгоритм ID3, запропонований Квінланом у 1986 році, використовує інформаційний приріст для вибору атрибутів розгалуження. Модифікація C4.5 вирішує проблему схильності до атрибутів з великою кількістю значень шляхом використання коефіцієнта інформаційного приросту. Алгоритм CART (Classification and Regression Trees) застосовує індекс Джині для класифікації та середньоквадратичну помилку для регресії, підтримуючи як категоріальні, так і числові атрибути [1].

Головною перевагою класичних дерев є їх інтерпретованість: кожне рішення може бути простежене через послідовність логічних умов. Проблема вибору оптимальної структури дерева залишається однією з ключових проблем у традиційних підходах. Методи прунінгу – pre-pruning та post-pruning – спрямовані на мінімізацію складності дерева при збереженні точності прогнозування. Разом з тим, у задачах з нелінійними залежностями, зашумленими даними або складними взаємодіями між ознаками дерева рішень поступаються більш виразним моделям [1].

1.2. Нейронні мережі та їх роль у машинному навчанні

На відміну від дерев рішень, нейронні мережі будують свої передбачення, імітуючи принципи роботи мозку. Вони складаються з великої кількості взаємопов'язаних нейронів – математичних функцій, що отримують вхідні дані, перетворюють їх і передають далі через мережу. Кожен нейрон вчиться виявляти певні шаблони або закономірності в даних, і чим більше таких рівнів (шарів), тим складніші залежності здатна розпізнати модель [3].

Нейронні мережі та глибоке навчання революціонізували машинне навчання завдяки можливості автоматичного вивчення представлень даних. Багатошарові персептрони використовують зворотне поширення похибки для навчання ваг синапсів. Сучасні архітектури, такі як згорткові та рекурентні мережі, демонструють гарні результати у комп'ютерному зорі, обробці природної мови та інших областях [3, 4].

Ключовою перевагою нейронних мереж є їх універсальна апроксимаційна здатність: теоретично, достатньо широка мережа може наблизити будь-яку неперервну функцію. Практично це означає можливість моделювання складних залежностей у даних без необхідності ручного конструювання ознак. Функція активації відіграє критичну роль у здатності мережі до навчання нелінійних залежностей; ReLU, сигмоїд, tanh та їх варіації забезпечують різні властивості градієнтного потоку.

Процес навчання нейронних мереж базується на мінімізації функції втрат через градієнтний спуск. Регуляризаційні техніки, такі як L1/L2-регуляризація чи dropout, запобігають перенавчанню та покращують узагальнюючу здатність моделі.

1.3. Гібридні моделі та їх класифікація

Гібридні моделі в машинному навчанні представляють спробу об'єднати переваги різних підходів. Ансамблі методів, такі як Random Forest та Gradient Boosting, поєднують множину простих моделей для досягнення кращої точності. Нейросимвольні підходи інтегрують логічне мислення з нейронними обчисленнями [6].

Специфічні гібриди дерев рішень та нейронних мереж можна класифікувати за способом інтеграції:

- послідовна інтеграція: нейронна мережа обробляє вихід дерева або навпаки;
- паралельна інтеграція: результати об'єднуються зваженою сумою;
- вбудована інтеграція: нейронні компоненти інтегровані безпосередньо у структуру дерева.

Кожен підхід має свої переваги та обмеження в контексті конкретних застосувань.

2. Сучасні моделі дерев рішень з нейронною підтримкою

2.1. Neural Decision Trees (NDT)

Neural Decision Trees (NDT) представляють піонерський підхід до створення диференційованих дерев рішень [7]. Основна ідея полягає у заміні дискретних розгалужень «м'якими» функціями, що дозволяє використовувати градієнтні методи оптимізації. Кожен внутрішній вузол дерева містить нейронну мережу, що обчислює ймовірність переходу до кожної з дочірніх гілок.

Архітектура NDT дозволяє кінцевому прогнозу бути зваженою комбінацією всіх листових вузлів, де ваги визначаються добутком ймовірностей проходження шляху від кореня до листа. Такий підхід забезпечує гладкість функції втрат та можливість ефективного навчання методом зворотного поширення.

Математично ймовірність досягнення листа l через шлях від кореня обчислюється як добуток ймовірностей у всіх проміжних вузлах:

$$P(1 \mid x) = \prod P(decision_i \mid x),$$

де $decision_i$ – рішення в i-му вузлі шляху.

Ключовою інновацією NDT є використання диференційованих функцій розщеплення. Замість дискретних порогових функцій кожен вузол використовує сигмоїдну функцію:

$$P(right \mid x) = \sigma(f(x)),$$

де $f(x)$ – лінійна або нелінійна функція вхідного вектора.

Така архітектура дозволяє навчати всі параметри дерева одночасно через стандартні методи оптимізації [7].

2.2. Differentiable Decision Trees

Differentiable Decision Trees розширюють концепцію NDT шляхом використання більш складних функцій розгалудження. Замість простих лінійних перетворень у вузлах застосовуються повноцінні нейронні мережі, здатні виявляти нелінійні залежності в даних. Особливу увагу приділено методам регуляризації, що запобігають деградації структури дерева до суцільної нейронної мережі.

Регуляризація структури дерева включає штрафи за складність окремих функцій розщеплення та загальну глибину дерева. Функція втрат має такий вигляд:

Під час виведення зразок проходить через дерево, починаючи з кореня, і на кожному кроці вибирається дочірній вузол з найбільшою косинусною подібністю до поточного зразка.

Ключовою інновацією NBDT є Tree Supervision Loss – спеціалізована функція втрат, що поєднує стандартну втрату класифікації з додатковим членом, який заохочує правильне прийняття рішень на всіх рівнях дерева. Це дозволяє моделі навчатися не лише фінальній класифікації, але й проміжним рішенням на шляху до неї.

Бібліотека підтримує два основні режими роботи:

- м'який вивід (soft-режим) – використовує ймовірнісну маршрутизацію, де зразок може «протікати» через множину шляхів з різними вагами; забезпечує кращу точність;
- жорсткий вивід (hard-режим) – застосовує детерміністичну маршрутизацію, вибираючи єдиний шлях на кожному кроці; надає більш чітку інтерпретацію.

NBDT демонструє такі результати на стандартних датасетах комп'ютерного зору: на CIFAR-10 досягається точність 97.55%, що на 3.23% краще за найкращі попередні методи поєднання дерев рішень та глибокого навчання; на CIFAR-100 покращення становить 6.73% з досягненням 82.97% точності; на ImageNet NBDT досягає 76.60% точності, що на 15.31% краще за попередні підходи [9]. Особливо важливою є здатність NBDT до узагальнення на незнайомі класи: експерименти показують покращення до 16% при класифікації зразків з класів, що не використовувалися під час навчання.

3. Компоненти гібридних дерев рішень та алгоритми навчання

3.1. М'які дерева рішень з диференційованими розгалуженнями

Традиційні дерева рішень працюють як жорстка система «так – ні»: дані у кожному вузлі направляються лише в одну гілку. Гібридні дерева рішень працюють інакше – вони дозволяють даним проходити через усі можливі шляхи одночасно, але з різними ступенями ймовірності.

Така архітектура має важливу перевагу: усі операції стають гладкими і диференційованими, що дозволяє використовувати стандартні методи машинного навчання для оптимізації моделі.

Нейронні мережі у вузлах дерева замінюють просту логіку порівняння повноцінними малими нейронними мережами (зазвичай 1-2 приховані шари з 32-128 нейронами), які здатні виявляти складні залежності між різними ознаками. Для запобігання перенавчанню використовуються: dropout у функціях розщеплення, обмеження складності параметрів та нормалізація даних для стабільності навчання.

Механізми уваги дозволяють дереву динамічно вибирати, на які ознаки звертати увагу для кожного конкретного зразка – аналогічно до того, як людина залежно від ситуації звертає увагу на різні аспекти проблеми. Механізм самоуваги дозволяє моделі аналізувати взаємодії між різними ознаками, що є особливо корисним для табличних даних з можливими складними залежностями між стовпцями. Багатополюсна увага дає моделі можливість одночасно фокусуватися на різних аспектах даних.

Ансамблі нейронних дерев замість використання одного дерева об'єднують кілька різних гібридних моделей для отримання більш точних та стабільних результатів. На відміну від традиційних лісів, що просто навчають багато схожих дерев на різних підвибірках даних, ансамблі нейронних дерев можуть використовувати різні архітектури, різні способи навчання чи різні критерії оптимізації. Методи поєднання прогнозів включають: просте усереднення, зважене усереднення та навчання спеціальної мета-моделі [6].

3.2. Алгоритми навчання гібридних моделей

Градієнтний спуск є основним інструментом навчання гібридних дерев рішень. Традиційні дерева рішень складно оптимізувати за допомогою градієнтних методів через їх дискретну природу: неможливо обчислити, як саме малі зміни параметрів вплинуть на результат. Гібридні моделі вирішують цю проблему, роблячи всі операції гладкими та диференційованими. Процес навчання полягає у мінімізації функції втрат, що вимірює різницю між прогнозами моделі та правильними відповідями, доповненої регуляризаційними членами. Схема процесу:

початкові параметри → прогноз → обчислення втрат → обчислення градієнтів → оновлення параметрів → повторення.

Обчислення градієнтів для гібридних дерев складніше, ніж для звичайних нейронних мереж, оскільки потрібно враховувати вплив змін в одному вузлі на всю структуру дерева. Стабільність градієнтів критично важлива – використовується обрізання градієнтів для запобігання різким стрибкам параметрів; адаптивні методи навчання, що автоматично підстроюють швидкість навчання, зазвичай працюють краще за стандартні підходи.

Регуляризація є необхідним компонентом навчання гібридних моделей, оскільки висока виразна потужність робить їх схильними до перенавчання. Основні типи регуляризації представлено у таблиці 1.

Таблиця 1

Основні типи регуляризації гібридних дерев рішень

Тип регуляризації	Опис	Застосування
Параметрична	Обмеження складності параметрів мереж	Штрафи за великі ваги
Структурна	Контроль архітектури дерева	Обмеження глибини, кількості вузлів
Стохастична	Випадкове відключення компонентів	Dropout у функціях розщеплення
Ентропійна	Заохочення визначених рішень	Штраф за невизначеність у вузлах

Рання зупинка навчання реалізується через відстеження якості моделі на валідаційних даних: якщо протягом певного часу покращення не спостерігається, навчання припиняється. Dropout у деревах рішень може застосовуватися по-різному: відключення цілих вузлів, окремих нейронів у функціях розщеплення чи випадкове блокування деяких шляхів проходження даних через дерево.

Прунінг – процес спрощення дерева через видалення найменш важливих частин. У гібридних моделях це складніше, ніж у традиційних деревах, оскільки всі вузли впливають на фінальний результат. Етапи прунінгу:

навчання повної моделі → оцінка важливості вузлів → поступове видалення → донавчання.

Методи оцінки важливості включають аналіз розмірів параметрів (менші параметри – менш важливі вузли), аналіз градієнтів (вузли з малими градієнтами менш важливі) та аналіз впливу на валідаційну втрату.

Оптимізація гіперпараметрів гібридних моделей передбачає налаштування таких параметрів, як глибина дерева, розмір нейронних мереж у вузлах, швидкість навчання, коефіцієнти регуляризації. Прості методи (пошук по сітці, випадковий пошук) прості у реалізації, але неефективні для складних просторів параметрів. Інтелектуальні методи (Байєсова оптимізація, еволюційні алгоритми) потребують меншої кількості обчислювальних ресурсів. Комбінований метод BOHB поєднує переваги різних підходів: ефективність Байєсової оптимізації зі швидкістю випадкового пошуку та раннього зупинення неперспективних конфігурацій.

4. Практична реалізація та результати

4.1. Середовище реалізації та інструментарій

У роботі розроблено серію демонстраційних програм мовою Python з використанням бібліотек scikit-learn (класифікатори MLP та DecisionTree), NumPy, matplotlib, shap та pandas. Для демонстрації бібліотеки NBDT використовується PyTorch та torchvision. Програми реалізовані з підвищенням складності від базової гібридної моделі до повноцінного аналізу з поясненням рішень.

4.2. Гібридна модель на датасеті Iris

Перший програмний приклад (програма snt1.py) демонструє підхід до створення гібридної моделі, що поєднує нейронну мережу та дерево рішень на датасеті Iris (150 зразків, 4 ознаки, 3 класи квіток).

Логіка побудови моделі:

1. Навчання нейронної мережі MLP (1 прихований шар з 10 нейронами, 1000 ітерацій).
2. Отримання вектора ймовірностей як нейронної підтримки: train_proba = mlp.predict_proba(X_train).
3. Розширення простору ознак: оригінальні 4 ознаки доповнюються 3 новими – ймовірностями для кожного класу, таким чином простір ознак розширюється з 4 до 7 вимірів: X_train_augmented = np.hstack([X_train, train_proba]).
4. Навчання дерева рішень на розширеному наборі ознак (max_depth=4).

Дерево рішень отримує доступ до ймовірностей від нейронної мережі як до додаткових ознак, що дозволяє йому використовувати узагальнену «думку» нейронної мережі при прийнятті рішень. Результатом є точність гібридної моделі 1.0 (100%) на тестовій вибірці. Висока точність пояснюється кількома чинниками. По-перше, датасет Iris є добре структурованим: вид Iris setosa практично лінійно відокремлений від двох інших видів. По-друге, гібридний підхід дозволяє об’єднати переваги двох принципово різних типів моделей: нейронна мережа відмінно справляється з виявленням складних нелінійних закономірностей, тоді як дерево рішень пропонує структурований, логічний підхід до класифікації. Розширення простору ознак створює багатший контекст – додаткові ознаки з нейронної мережі особливо корисні у випадках, коли зразок знаходиться на межі між двома класами за фізичними характеристиками.

4.3. Візуалізація гібридного дерева рішень

Другий програмний приклад (програма snt2.py) є розширенням попереднього та додає повноцінну візуалізацію структури дерева рішень засобами функції plot_tree бібліотеки matplotlib. Для цього формується список назв розширеного простору ознак: оригінальні 4 назви (sepal length, sepal width, petal length, petal width) та три нових (NN_proba_setosa, NN_proba_versicolor, NN_proba_virginica).

Таблиця 2

Порівняння точності базових методів та гібридної моделі на датасеті Iris

Модель	Опис	Точність на тестовій вибірці
MLP	1 прихований шар, 10 нейронів	~97%
Дерево рішень	max_depth=4, 4 ознаки	~97%
Гібридна модель	Дерево з нейронною підтримкою, 7 ознак	100%

Графічна репрезентація дерева (рис. 1) розкриває внутрішню логіку гібридної моделі. Кольорове кодування вузлів надає миттєву візуальну інформацію про чистоту класифікації в кожному розгалуженні. У вузлах дерева присутні як оригінальні ознаки (sepal width (cm)), так і нейронні ймовірності (NN_proba_virginica <= 0.016, NN_proba_versicolor <= 0.504), що підтверджує активне використання нейронної підтримки при прийнятті рішень. Умови типу NN_proba_setosa <= 0.001 демонструють, що дерево активно використовує висновки нейронної мережі як повноцінні ознаки для класифікації.

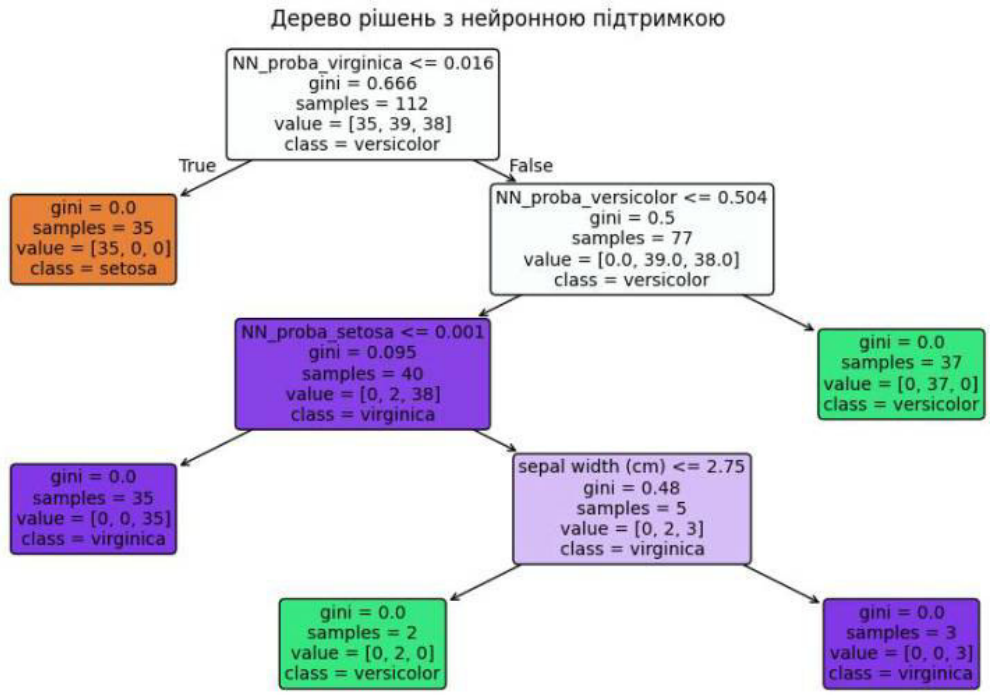


Рис. 1. Візуалізація дерева рішень з нейронною підтримкою

4.4. SHAP-аналіз для інтерпретації рішень

Третій програмний приклад (програма snt3.py) додає до гібридної моделі можливості пояснюваного штучного інтелекту (Explainable AI) через інтеграцію бібліотеки SHAP (SHapley Additive exPlanations).

SHAP-аналіз виявляє нові можливості розуміння поведінки гібридної моделі, надаючи кількісні метрики впливу кожної ознаки на процес прийняття рішень. Особливо цінним є аналіз взаємодій між оригінальними фізичними ознаками та нейронними ймовірностями: у деяких випадках нейронні ймовірності мають вищий вплив на рішення, ніж оригінальні ознаки, що свідчить про ефективність гібридного

підходу; в інших – традиційні ознаки домінують, що вказує на ситуації, де пряме вимірювання є більш інформативним за складні нейронні обчислення.

Для аналізу дерев рішень застосовується shap.TreeExplainer, що використовує ефективні алгоритми для точного обчислення значень Шеплі.

Метод shap_values повертає матрицю, де кожен елемент показує, наскільки конкретна ознака збільшує або зменшує ймовірність віднесення зразка до конкретного класу. Функція summary_plot створює комплексну візуалізацію (рис. 2), що поєднує інформацію про важливість ознак з їх значеннями для всіх зразків тестової вибірки.

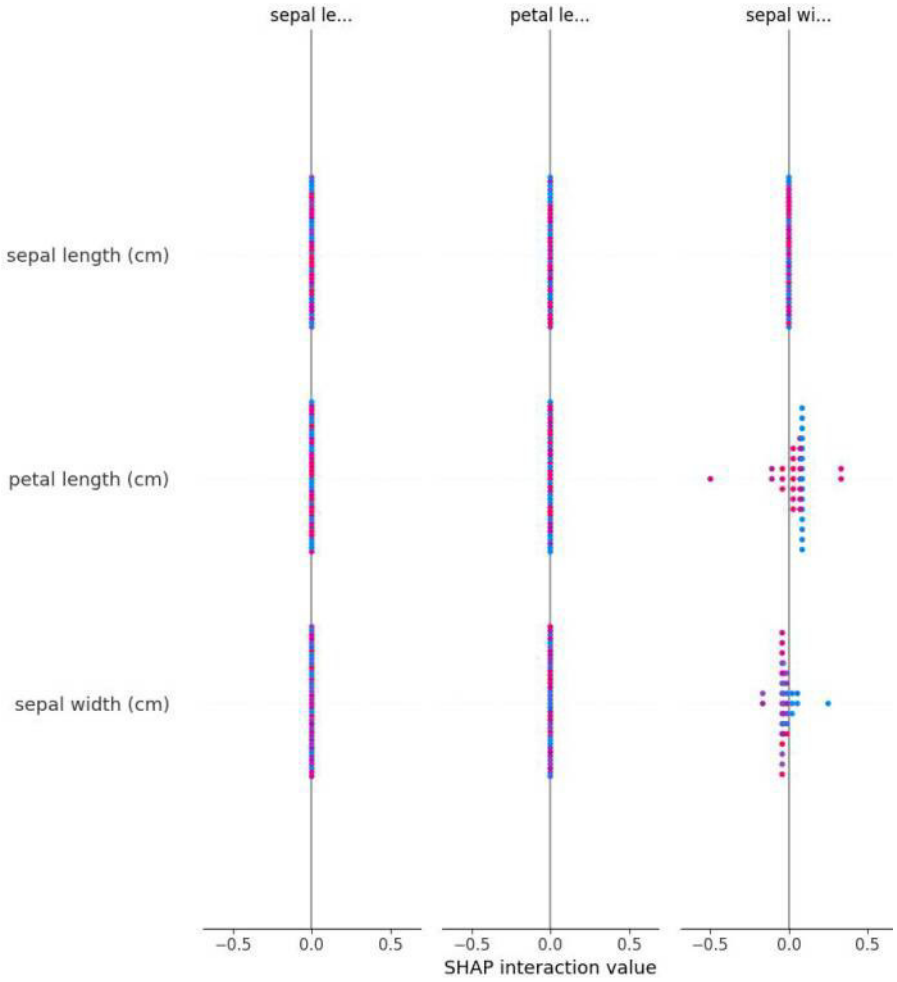


Рис.2. Візуалізація результатів SHAP-аналізу

Кольорове кодування на Summary Plot розкриває нелінійні залежності між значеннями ознак та їх впливом на класифікацію: висока довжина пелюстки може мати позитивний SHAP-вплив для класу virginica, але негативний для setosa. Загальна картина SHAP-аналізу створює основу для подальшого вдосконалення гібридної моделі, надаючи конкретні, кількісно обґрунтовані рекомендації щодо того, які аспекти моделі працюють найефективніше.

4.5. Застосування до реальних даних: датасет нобелівських лауреатів

Четвертий програмний приклад (програма snt4.py) демонструє перехід від класичного датасету Iris до практичного застосування гібридної моделі на реальних даних про нобелівських лауреатів. Для роботи з CSV-файлом використовується бібліотека pandas.

Особливістю реалізації є створення нових ознак зі стовпця дати народження: обчислюється рік народження та похідна ознака «вік лауреата на момент отримання премії». Категоріальні ознаки (стать, країна народження) кодуються за допомогою LabelEncoder. Числові ознаки масштабуються за допомогою StandardScaler, що є критично важливим для нейронної мережі при роботі з ознаками різного масштабу (рік може бути 1900–2020, тоді як стать кодується як 0 або 1). Цільова змінна – категорія нобелівської премії, закодована також через LabelEncoder.

Простір ознак гібридної моделі формується динамічно залежно від кількості категорій премій: ["sex", "country", "year", "age"] + [f"NN_proba_{i}" for i in range(train_proba.shape)] [8].

Точність моделі на цьому датасеті є нижчою порівняно з Iris через значно більшу складність та суб'єктивність процесу присудження нобелівських премій: рішення комітетів залежать від багатьох факторів, які не відображені в базових демографічних даних, що робить задачу класифікації значно складнішою. Тим не менш гібридна модель стабільно перевершує базові методи, а дерево рішень виявляє цікаві закономірності – наприклад, щодо ролі нейронних ймовірностей у першому вузлі дерева та взаємодії демографічних ознак із категоріями премій.

5. Аналіз результатів

Результати порівняльного аналізу розглянутих гібридних архітектур наведено в таблиці 3.

Таблиця 3

Порівняльний аналіз розглянутих гібридних архітектур

Архітектура	Основна ідея	Тип розщеплення	Інтерпрето-ваність	Ефективність на табл. даних
NDT [4]	М'які розгалуження	Лінійна сигмоїдна	Середня	Середня
Differentiable DT	Нейромережі у вузлах	Нелінійна нейромережа	Середня	Висока
NODE [11]	Oblivious-структура	Entmax-зважена	Низька	Висока
TabNet [10]	Sequential attention	Attention-маски	Висока	Висока
NBDT [9]	Ієрархія з ваг FC-шару	Косинусна подібність	Висока	Середня
Нейронна підтримка	Ймовірності MLP як ознаки	Стандартна (CART)	Висока	Середня

Порівняння продуктивності показує, що гібридні моделі особливо ефективні на датасетах середнього розміру, де ансамблеві методи можуть страждати від перенавчання, а прості моделі – від недостатньої виразності. Ансамблі дерев мають переваги у стійкості через агрегацію множини моделей, однак це призводить до значно більшої обчислювальної складності як під час навчання, так і під час виведення.

Гібридні моделі пропонують компромісне рішення, поєднуючи виразність складних моделей з відносною простотою одиничної архітектури [11].

Висновки

У роботі проведено комплексне дослідження гібридних моделей машинного навчання, що поєднують дерева рішень із нейронними мережами.

Встановлено, що традиційні дерева рішень і нейронні мережі мають взаємодоповнювальні властивості, а гібридні архітектури ефективно поєднують переваги обох підходів. Розглянуто сучасні гібридні архітектури (NDT, Differentiable Decision Trees, NODE, TabNet, NBDT) та виділено їх ключові особливості з точки зору типу розщеплення, навчальності та інтерпретованості.

Практично реалізовано та верифіковано метод нейронної підтримки: на датасеті Iris гібридна модель досягла точності 100%, перевершивши обидва базових методи. Засобами SHAP-аналізу підтверджено, що нейронні ймовірності є значущими предикторами для дерева рішень і відіграють вирішальну роль у вузлах з високою невизначеністю. Застосування методу до датасету нобелівських лауреатів демонструє його практичну застосовність на реальних соціальних даних.

Розглянута бібліотека NBDT підтверджує, що описаний принцип масштабується до задач комп'ютерного зору (97.55% на CIFAR-10, 82.97% на CIFAR-100, 76.60% на ImageNet), що свідчить про широкий потенціал гібридних нейронно-символьних підходів.

Подальші дослідження доцільно спрямувати на аналіз поведінки запропонованих моделей при підвищенні розмірності та зашумленості даних, а також на дослідження методів автоматичного визначення оптимального способу інтеграції нейронної та символьної компонент.

Список використаної літератури:

1. Breiman L., Friedman J., Stone C. J., Olshen R. A. Classification and regression trees. – CRC press, 1984.
2. Quinlan J. R. Induction of decision trees // Machine learning. – 1986. – Vol. 1 (1). – P. 81–106.
3. Goodfellow I., Bengio Y., Courville A. Deep learning. – MIT press, 2016. – 800 p.
4. LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. – 2015. – Vol. 521 (7553). – P. 436–444.
5. Kontschieder P., Fiterau M., Criminisi A., Rota Bulo S. Deep neural decision forests // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2015. – P. 1467–1475.
6. Zhou Z. H. Ensemble methods: foundations and algorithms. – CRC press, 2012.
7. Irsoy O., Yıldız O. T., Alpaydın E. Soft decision trees // In 21st International Conference on Pattern Recognition (ICPR). – 2012. – P. 1819–1822.
8. Kaddour J., Lynch A., Liu Q., Kusner M. J., Silva R. When do neural nets outperform boosted trees on tabular data? // arXiv preprint arXiv:2305.02997. – 2022.
9. Neural-Backed Decision Trees [Електронний ресурс]. – Режим доступу: https://research.alvinwan.com/neural-backed-decision-trees. – Назва з екрана.
10. Arik S. Ö., Pfister T. TabNet: Attentive interpretable tabular learning // Proceedings of the AAAI Conference on Artificial Intelligence. – 2021. – Vol. 35, No. 8. – P. 6679–6687.
11. Popov S., Morozov S., Babenko A. Neural oblivious decision trees for deep learning on tabular data // arXiv preprint arXiv:1909.06312. – 2019.

References:

1. Breiman L., Friedman J., Stone C. J., Olshen R. A. (1984) Classification and regression trees. CRC press.
2. Quinlan J. R. (1986) Induction of decision trees. Machine learning, 1 (1), pp. 81–106.
3. Goodfellow I., Bengio Y., Courville A. (2016) Deep learning. MIT press.
4. LeCun Y., Bengio Y., Hinton G. (2015) Deep learning. Nature, 521 (7553), pp. 436–444.
5. Kontschieder P., Fiterau M., Criminisi A., Rota Bulo S. (2015) Deep neural decision forests. Proceedings of the IEEE ICCV, pp. 1467–1475.
6. Zhou Z. H. (2012) Ensemble methods: foundations and algorithms. CRC press.
7. Irsoy O., Yıldız O. T., Alpaydın E. (2012) Soft decision trees. 21st International Conference on Pattern Recognition (ICPR), pp. 1819–1822.

8. Kaddour J., Lynch A., Liu Q., Kusner M. J., Silva R. (2022) When do neural nets outperform boosted trees on tabular data? arXiv:2305.02997.
9. Neural-Backed Decision Trees. Available at: https://research.alvinwan.com/neural-backed-decision-trees/
10. Arik S. Ö., Pfister T. (2021) TabNet: Attentive interpretable tabular learning. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 8, pp. 6679–6687.
11. Popov S., Morozov S., Babenko A. (2019) Neural oblivious decision trees for deep learning on tabular data. arXiv:1909.06312.

PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Head of Department of Applied Mathematics and Informatics, Bohdan Khmelnytsky National University of Cherkasy
KRASNOSHLYK Nataliya,
Candidate of Technical Sciences, Associate Professor, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
SVIRENKO Danylo
Software Developer, Cherkasy, Ukraine

APPLICATION OF NEURAL-ASSISTED DECISION TREES IN ARTIFICIAL INTELLIGENCE PROBLEMS

Summary. Introduction.

The article investigates hybrid machine learning models that combine decision trees with neural networks to simultaneously achieve high prediction accuracy and interpretability of decisions. The modern architectures of the following models are analyzed: Neural Decision Trees (NDT), Differentiable Decision Trees, Neural Oblivious Decision Trees (NODE), TabNet, and Neural-Backed Decision Trees (NBDT). The practical implementation includes building a hybrid model on the Iris and Nobel laureate datasets using SHAP analysis to interpret the results, which confirms the effectiveness and practical applicability of the described approach.

Modern machine learning faces the fundamental challenge of balancing predictive accuracy with model interpretability. Classical decision trees offer transparent, rule-based reasoning but are limited in capturing complex nonlinear patterns. Neural networks achieve state-of-the-art accuracy but function as opaque "black boxes." Hybrid models that combine the strengths of both approaches represent a promising research direction, particularly for safety-critical applications where algorithmic decisions must be explainable.

Purpose. The aim of this article is to investigate the theoretical foundations and modern architectures of hybrid models combining decision trees with neural networks, to practically implement such models on classification tasks, and to evaluate the interpretability of obtained decisions using SHAP analysis.

Results. The paper systematizes and compares five classes of modern hybrid architectures: Neural Decision Trees (NDT), Differentiable Decision Trees, Neural Oblivious Decision Trees (NODE), TabNet, and Neural-Backed Decision Trees (NBDT). The components of hybrid decision trees are examined in detail: soft differentiable branching, neural networks as split functions, attention mechanisms, and ensemble approaches. Training algorithms – gradient descent, regularization types (parametric, structural, stochastic, and entropy-based), pruning, and hyperparameter optimization – are described. The NBDT library achieves 97.55% on CIFAR-10, 82.97% on CIFAR-100, and 76.60% on ImageNet, surpassing previous hybrid methods by 3.23%, 6.73%, and 15.31% respectively. A two-stage hybrid model is implemented, where the probabilistic outputs of an MLP are used as additional features for a decision tree. The hybrid model achieves 100% accuracy on the Iris dataset, outperforming both baseline models. SHAP analysis confirms that neural probability features play a decisive role in uncertain nodes, while original features dominate in straightforward cases. Application to the Nobel laureates dataset further validates the approach on real social data.

Conclusion. Hybrid models combining decision trees with neural network support effectively resolve the accuracy–interpretability trade-off. The proposed neural support mechanism enriches the feature space of a decision tree without sacrificing its structural interpretability, as confirmed through experiments on two different datasets and quantitative SHAP-based explanations.

Keywords: decision trees, neural networks, hybrid models, soft decision trees, NBDT, SHAP, machine learning, interpretability.

Одержано редакцією 05.09.2024 р.
Прийнято до публікації 30.10.2024 р.

УДК 004.415.2:004.5

DOI 10.31651/2076-5886-2024-1-79-90

PACS 07.05.Tp, 89.20.Ff

ВОЙЦІХОВСЬКА Лєна Іванівна
студентка спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького

ДЗЮБА Вікторія Анатоліївна
кандидат технічних наук, старший викладач кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: viktoriya.dzyuba15@vu.cdu.edu.ua
ORCID 0000-0003-1655-0333

РОЗРОБКА ІНФОРМАЦІЙНО-ТЕСТУВАЛЬНОГО ВЕБ-ДОДАТКУ ДЛЯ НАВЧАЛЬНОГО ЗАКЛАДУ

У статті розглядається процес проєктування, розробки та розгортання сучасного веб-застосунку для навчальної платформи. Детально аналізується архітектура системи, включаючи бази даних PostgreSQL, використання ORM Prisma, а також побудова серверної частини на основі Node.js, TypeScript та фреймворку NestJS. Описано принципи організації REST API, механізми аутентифікації та авторизації користувачів із використанням JWT, а також підходи до безпечного зберігання даних і управління доступом.

У роботі також висвітлено розробку клієнтської частини застосунку із використанням React, Vite, Redux та бібліотеки MUI, що дозволило створити інтерактивний і масштабований інтерфейс користувача. Окрему увагу приділено процесу проєктування інтерфейсу у Figma та практичній реалізації системи розгортання за допомогою Docker, MinIO для зберігання файлів і MailHog для тестування електронної пошти. Результатом роботи є комплексне програмне рішення, яке поєднує сучасні технології веб-розробки та забезпечує стабільність, безпеку й зручність використання

.***Ключові слова:*** *веб-застосунок, інформаційна система, клієнт-серверна архітектура, база даних, PostgreSQL, Prisma, Node.js, TypeScript, NestJS, REST API, JWT, аутентифікація, авторизація, React, SPA, Redux, Vite, MUI, Figma, Docker, MinIO, MailHog, розгортання, веб-розробка.*

Вступ

Сучасний ритм життя вносить свої корективи в наші плани та буденність, змушуючи змінювати тимчасово чи на зовсім своє місце проживання, перебувати по декілька годин в укритті, правильно виставляти пріоритети та економити електроенергію. Увесь світ, зокрема Україна, проходить зараз етап цифровізації, щоб будь-яка людина змогла зробити базові речі, не виходячи з дому. Це покупки в магазині, оформлення документів, запис до певного спеціаліста і тому подібне. Таким чином, у деякій мірі, можна забезпечити вдале поєднання безпеки та комфорту людського життя.

Цифровізації зазнає і освіта, адже у такому потоці хаосу потрібно продовжувати навчатись та розвиватись, незалежно від розташування та речей поруч. Для цього

потрібно, щоб все необхідне було в одному місці. Звичайно, технології зараз не замінять вчителя, однак додаткові матеріали, домашні завдання, оцінки - це все, що розміщено на одному сайті з доступом для батьків, учнів та вчителів, спростить життя та додасть мобільності освітньому процесу. Такий підхід буде економією часу та ресурсів у вільному доступі для будь-якого навчального закладу чи організації.

Мета статті – створення інформаційно-тестувального веб-додатку для навчального закладу з базовим та найбільш необхідним функціоналом, який на основі досвіду користувачів, підвищенні власної кваліфікації та розширення команди можна буде вдосконалювати надалі.

Виклад основного матеріалу

1. Аналіз предметної області та постановка задачі

Сучасний етап цифровізації характеризується стрімким проникненням інформаційних технологій у всі сфери суспільного життя, зокрема в освіту. Поширення систем дистанційного та змішаного навчання зумовило появу великої кількості освітніх платформ, що забезпечують організацію навчального процесу в онлайн-середовищі. У межах даного дослідження розглянуто три найбільш поширені рішення: Google Classroom, «Єдина Школа» та освітню платформу Optima.

Google Classroom є безкоштовним сервісом компанії Google, призначеним для організації навчального процесу, створення та перевірки завдань, а також комунікації між викладачами та учнями. Основною перевагою системи є глибока інтеграція з екосистемою Google (Docs, Sheets, Slides, Forms), що забезпечує зручність роботи без використання сторонніх програмних засобів. Платформа має інтуїтивний інтерфейс і не потребує складного налаштування, що робить її доступною для широкого кола користувачів.

«Єдина Школа» – це комплексна інформаційна система для закладів освіти, яка включає електронний журнал, щоденник, засоби комунікації між учасниками освітнього процесу та функціонал дистанційного навчання. Система орієнтована на офіційно зареєстровані навчальні заклади та інтегрується з державними освітніми реєстрами. Її основною особливістю є централізоване управління навчальним процесом, проте недоліком є складність впровадження та необхідність адміністративної інтеграції закладу до системи, а також платна модель використання після пробного періоду.

Optima є комерційною платформою дистанційної освіти для учнів 1-11 класів. Вона забезпечує повноцінний навчальний цикл, включаючи доступ до навчальних матеріалів, відеоуроків, інтерактивних завдань і тестів. Перевагою є високий рівень інтерактивності та методичної структурованості навчального контенту. Водночас суттєвим недоліком є висока вартість навчання, що обмежує доступність платформи для широкої аудиторії.

Порівняльний аналіз зазначених систем показує, що всі вони забезпечують базовий набір функцій, необхідних для організації навчального процесу: створення класів, завдань, тестів та автоматизоване оцінювання. Проте кожна система має власні обмеження, пов’язані або з закритістю екосистеми, або з фінансовою доступністю, або з обмеженою гнучкістю впровадження.

Основними вимогами до сучасної освітньої системи є: зручна та передбачувана навігація, адаптивна архітектура інтерфейсу, розподіл ролей користувачів та чітке розмежування прав доступу. Проєктована система передбачає чотири основні ролі користувачів: учень, вчитель, директор та адміністратор. Для кожної ролі визначено відповідний набір дозволів, що регламентує доступ до функціональних модулів платформи. Окремо виділяється категорія незареєстрованих користувачів (гостей), для

яких передбачено обмежений доступ до інформаційних розділів, зокрема блогу, контактної інформації та загальних відомостей про навчальні заклади.

Розробка програмного продукту розглядається як комплексний процес створення, впровадження та супроводу програмного забезпечення. У загальному вигляді життєвий цикл розробки можна поділити на такі етапи:

1. Аналіз предметної області та дослідження ринку
2. Формування вимог до системи
3. Технічне проєктування
4. Безпосередня розробка програмного забезпечення
5. Тестування та оцінка якості

Перший етап є критично важливим, оскільки визначає архітектуру майбутньої системи та впливає на всі наступні стадії. На основі зібраних вимог формується технічне завдання, яке визначає функціональність системи, вимоги до інтерфейсу та бізнес-логіки.

Технічне проєктування включає розробку дизайну інтерфейсу, структури бази даних та логіки взаємодії компонентів. Етап розробки передбачає налаштування середовища, підключення необхідних технологій та реалізацію функціоналу. Завершальними стадіями є тестування, рефакторинг та оптимізація коду. У процесі розробки програмної системи використовуються сучасні інструменти, що забезпечують ефективність командної роботи та якість кінцевого продукту.

Для управління задачами застосовано Trello, що дозволяє структурувати процес розробки та відстежувати виконання завдань. Проєктування інтерфейсу здійснювалося у середовищі Figma, яке є стандартом у сфері UI/UX дизайну та дозволяє створювати інтерактивні макети.

Основним середовищем розробки було обрано Visual Studio Code, яке забезпечує гнучкість, підтримку розширень та зручну роботу з кодом. Контроль версій реалізовано за допомогою Git, що дозволяє ефективно керувати змінами та організовувати командну розробку.

Для роботи з базою даних використано pgAdmin як інструмент адміністрування PostgreSQL, що забезпечує зручний графічний інтерфейс для управління даними. Для документування та тестування API застосовано Swagger, який дозволяє формалізувати взаємодію між клієнтською та серверною частинами системи.

2. Створення моделей, розробка логіки та серверної частини продукту

Еволюція систем керування базами даних (СКБД) відображає загальний розвиток інформаційних технологій – від файлових структур до сучасних реляційних та гібридних моделей. Первинні підходи до зберігання даних ґрунтувалися на використанні файлів, що призводило до тісного зв’язку між програмним кодом і фізичною організацією даних, ускладнюючи масштабування та повторне використання інформації. Подальший розвиток призвів до появи централізованих систем керування файлами, однак їх обмеження (відсутність гнучкості, дублювання даних, складність спільного доступу) зумовили формування концепції баз даних як централізованого сховища взаємопов’язаних даних із керованим доступом. Важливим етапом стало впровадження реляційної моделі, яка забезпечила формалізований підхід до структурування даних і стала основою сучасних СКБД.

Проєктування бази даних у межах даної роботи здійснювалося на основі комбінованого підходу: з урахуванням як бізнес-вимог системи, так і особливостей предметної області. На етапі логічного моделювання було сформовано набір сутностей, що відображають ключові компоненти освітньої платформи: користувачі, ролі, школи,

класи, предмети, завдання, тести, питання, відповіді, файли, запрошення та публікації (рис. 1).

Для моделювання структури бази даних використано онлайн-інструмент SqlDBM, що дозволив візуалізувати сутності та зв'язки між ними. Такий підхід забезпечив цілісність моделі та спростив подальшу реалізацію фізичної структури бази даних.

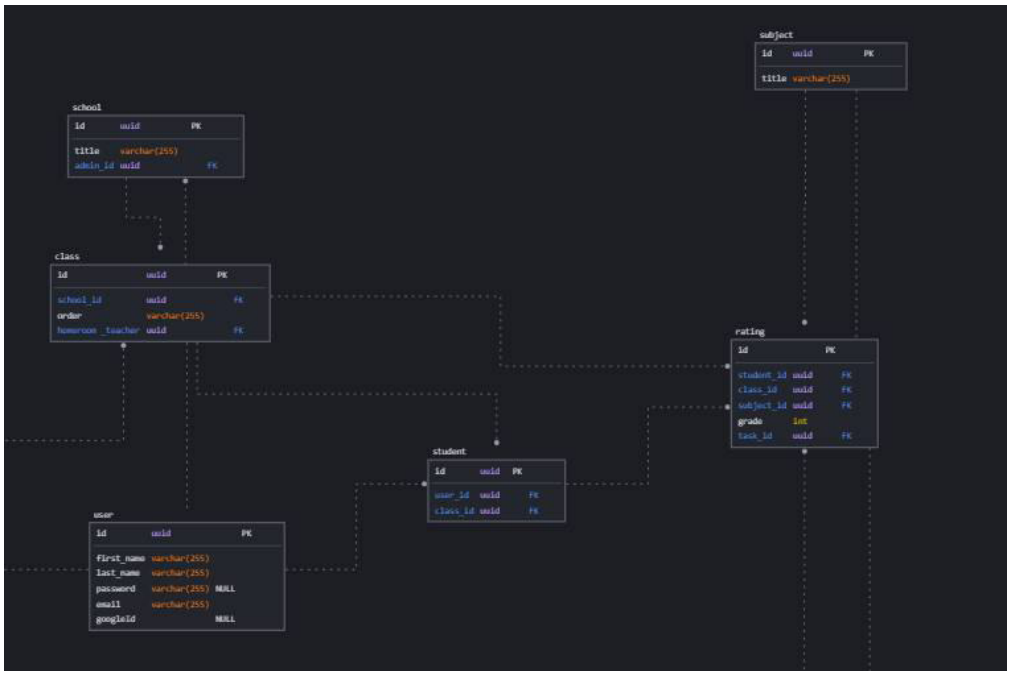


Рис. 1. Проектування моделей бази даних у SqlDBM

З урахуванням характеру системи, яка передбачає значну кількість взаємопов'язаних сутностей і транзакцій, обрано реляційну СКБД PostgreSQL. Вона забезпечує високу надійність, підтримку складних зв'язків, транзакційність та відповідність стандартам SQL.

PostgreSQL є відкритою, кросплатформною системою, що підтримує розширюваність через власні типи даних, функції та модулі (зокрема PostGIS). Її архітектура дозволяє ефективно працювати як з невеликими, так і з масштабними наборами даних, забезпечуючи стабільність і цілісність інформації. Таким чином, вибір PostgreSQL обґрунтовано необхідністю забезпечення складних зв'язків між сутностями та високими вимогами до цілісності даних у навчальній платформі.

Для адміністрування PostgreSQL використано pgAdmin, який надає графічний інтерфейс для створення, редагування та моніторингу об'єктів бази даних (рис. 2). Інструмент дозволяє виконувати SQL-запити, аналізувати продуктивність та керувати користувацькими правами доступу. Використання pgAdmin значно спрощує процес взаємодії з базою даних, особливо на етапі розробки та тестування, забезпечуючи зручність контролю структури та даних.

Для організації взаємодії між серверною частиною та базою даних використано ORM Prisma, який реалізує об'єктно-реляційне відображення даних. Основною перевагою ORM є можливість роботи з даними у вигляді об'єктів, без необхідності прямого написання SQL-запитів. Prisma забезпечує декларативне визначення моделей, автоматичну генерацію запитів, типізований доступ до даних та систему міграцій. Це підвищує надійність коду та зменшує ризик помилок при роботі з базою даних. У межах проєкту Prisma використовується для опису моделей, управління схемою бази даних та генерації типізованого API доступу до даних.

Серверна частина є ключовим компонентом системи, оскільки забезпечує обробку запитів, бізнес-логіку та взаємодію з базою даних. Основна модель роботи базується на принципі клієнт–серверної архітектури, де клієнт надсилає HTTP-запити, а сервер повертає структуровані відповіді.

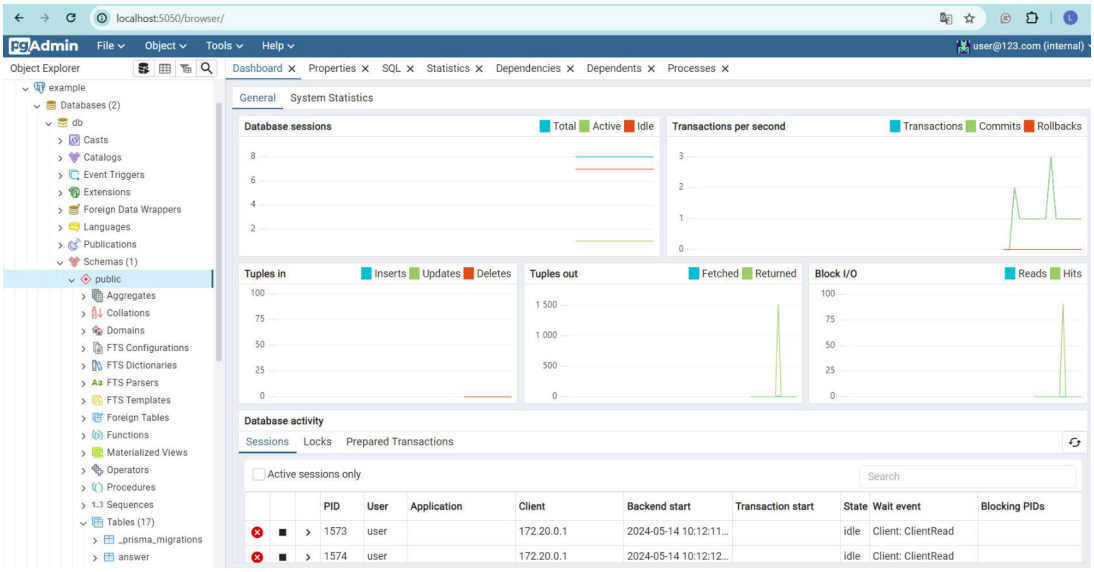


Рис. 2. Інтерфейс середовища PGAdmin

HTTP-протокол передбачає використання основних методів взаємодії з ресурсами: GET, POST, PUT, PATCH, DELETE, кожен з яких відповідає певному типу операцій. Результати обробки запитів визначаються через HTTP-коди стану, які відображають успішність або помилки виконання операцій. Конфігураційні параметри системи (порти, ключі доступу, адреси сервісів) зберігаються у змінних середовища (.env), що підвищує безпеку та гнучкість розгортання.

Для реалізації серверної частини використано Node.js, що дозволяє виконувати JavaScript поза браузером. Його ключовою перевагою є асинхронна неблокуюча модель обробки запитів, що забезпечує високу продуктивність при великій кількості одночасних з'єднань. Node.js широко використовується для побудови REST API, real-time застосунків та масштабованих веб-сервісів, що робить його оптимальним вибором для сучасних серверних рішень.

Для підвищення надійності коду використано TypeScript, який додає статичну типізацію до JavaScript. Це дозволяє зменшити кількість помилок на етапі розробки та забезпечує чітке визначення структур даних. У проєкті реалізовано типізацію моделей та API-запитів, що забезпечує узгодженість між серверною та клієнтською частинами системи.

Основою серверної архітектури виступає NestJS, який забезпечує модульну структуру застосунку. Архітектура NestJS базується на поділі на:

– контролери (обробка HTTP-запитів),
– сервіси (бізнес-логіка),
– модулі (логічне групування компонентів).

Такий підхід забезпечує масштабованість, тестованість та підтримуваність коду. Вхідною точкою системи є файл main.ts, який ініціалізує додаток і підключає всі модулі.

Для тестування API використано Postman, що дозволяє перевіряти роботу серверних маршрутів, параметрів запитів і відповідей. Документування API реалізовано за допомогою Swagger, який автоматично генерує інтерактивну документацію. Це

значно спрощує взаємодію між бекендом і фронтендом та зменшує необхідність ручного аналізу коду.

Безпека користувацьких даних реалізується через механізми аутентифікації та авторизації. Аутентифікація передбачає перевірку облікових даних користувача, тоді як авторизація визначає рівень доступу до ресурсів системи. Основою безпеки виступає модель ролей, відповідно до якої користувач отримує доступ лише до дозволених функцій.

Для реалізації автентифікації використано JWT (JSON Web Token, рис. 3). Токен складається з трьох частин: header, payload і signature. Він використовується для безпечної передачі даних між клієнтом і сервером. JWT зберігається на стороні клієнта у cookies та передається з кожним запитом. Перевірка підпису дозволяє підтвердити цілісність і достовірність даних.

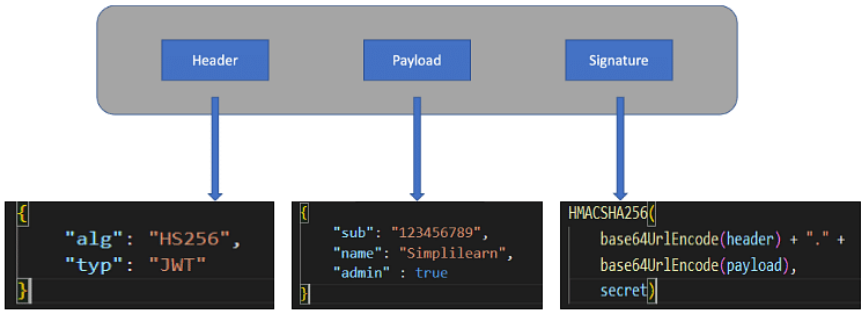


Рис. 3. Структура JSON Web токена.

Додатково для захисту паролів використовується хешування на основі bcrypt, що забезпечує одностороннє шифрування та захист від атак перебором.

Для реалізації автоматичної розсилки електронних повідомлень використано бібліотеку Nodemailer. Вона застосовується для:

– підтвердження реєстрації адміністратора,
– запрошення викладачів до закладу,
– запрошення учнів до класів.

Механізм базується на передачі токенізованих посилань через email, що дозволяє автоматизувати процес підключення користувачів до системи.

3. Дизайн та розробка клієнтської частини проєкту

Процес розробки клієнтської частини веб-застосунку розпочинається з проєктування інтерфейсу користувача. У межах даної роботи використано інструмент Figma, який є сучасним хмарним середовищем для створення UI/UX-дизайну та спільної роботи над макетами.

Figma забезпечує можливість швидкого прототипування інтерфейсів, узгодження дизайну між учасниками проєкту та реалізації компонентного підходу до побудови сторінок. Основними перевагами інструмента є кросплатформеність, підтримка колективної розробки та інтеграція з сучасними процесами UI/UX-дизайну. У межах проєкту в Figma було розроблено структуру інтерфейсу, логотип, іконографіку та головні сторінки системи, що дозволило сформувати цілісну візуальну концепцію майбутнього веб-застосунку (рис. 4).

Для побудови сучасного фронтенд-застосунку використано інструмент збірки Vite, який є одним із найсучасніших рішень для швидкої розробки веб-проєктів. На відміну від традиційних бандлерів (Webpack, Parcel), Vite використовує нативну

підтримку ES-модулів у браузері, що забезпечує значне прискорення запуску проєкту та оновлення коду. Його архітектура включає dev-сервер із підтримкою швидкого Hot Module Replacement (HMR) та оптимізовану production-збірку на основі Rollup. Застосування Vite дозволяє суттєво скоротити час розробки та підвищити продуктивність роботи над інтерфейсом, особливо у великих SPA-застосунках.

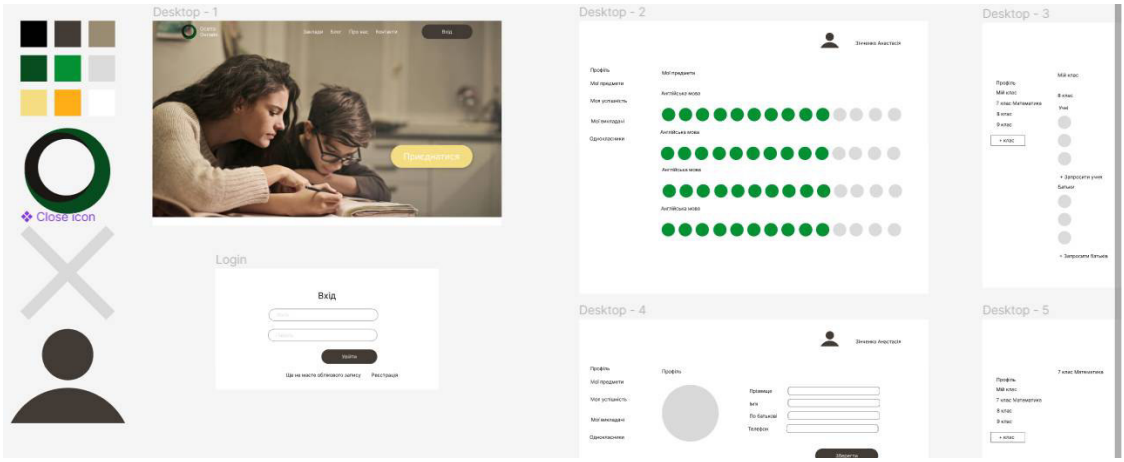


Рис. 4. Створення дизайну в Figma

Клієнтська частина реалізована як SPA (Single Page Application), що передбачає завантаження одного HTML-документа з подальшим динамічним оновленням контенту без повного перезавантаження сторінки. Такий підхід забезпечує:

– високу швидкість взаємодії з користувачем;
– зменшення навантаження на сервер;
– покращений користувацький досвід.

Для реалізації SPA використано бібліотеку ReactJS, яка базується на компонентній архітектурі. Основними перевагами React є:

– декларативний підхід до побудови UI;
– компонентна структура інтерфейсу;
– використання віртуального DOM для ефективного оновлення сторінок.

React дозволяє будувати масштабовані інтерфейси з повторно використовуваних компонентів, що є критично важливим для складних освітніх платформ.

Для опису інтерфейсу використовується JSX (JavaScript XML) – синтаксичне розширення JavaScript, що дозволяє поєднувати логіку та розмітку в межах одного файлу. JSX спрощує створення динамічних інтерфейсів, дозволяючи вбудовувати JavaScript-вирази безпосередньо в розмітку.

Для побудови інтерфейсних компонентів застосовано бібліотеку MUI (Material UI), яка реалізує принципи Material Design. MUI надає готові UI-компоненти, що дозволяє:

– прискорити розробку інтерфейсу;
– забезпечити уніфікований стиль системи;
– реалізувати адаптивність під різні пристрої.

Поєднання JSX і MUI забезпечує гнучкість у розробці складних інтерфейсів та дозволяє швидко формувати сучасний UI без значних витрат на кастомну стилізацію. У складних SPA-застосунках ключовою задачею є керування станом інтерфейсу. Для цього використано бібліотеку Redux, яка забезпечує централізоване управління станом застосунку.

Основні принципи Redux:

– єдине джерело стану (store);
– зміна стану лише через actions;
– використання pure reducers для обробки змін.

Такий підхід забезпечує передбачуваність поведінки застосунку та спрощує налагодження і масштабування. Інтеграція з React здійснюється через react-redux, що дозволяє компонентам отримувати доступ до стану через хуки useSelector і змінювати його через useDispatch. Використання Redux особливо ефективне у системах з великою кількістю взаємопов'язаних компонентів, де необхідна синхронізація даних між різними частинами інтерфейсу.

4. Збірка проєкту

Сучасна розробка веб-застосунків передбачає використання контейнеризації як базового підходу до розгортання програмного забезпечення. У межах даної роботи застосовано технологію Docker, яка дозволяє ізолювати сервіси в окремі контейнери разом із усіма залежностями. Контейнеризація забезпечує відтворюваність середовища, що є критично важливим для командної розробки та перенесення системи між різними середовищами (локальним, тестовим і продакшн). На відміну від віртуальних машин, Docker-контейнери є легшими, швидше запускаються та ефективніше використовують ресурси системи.

У межах проєкту було реалізовано багатоконтейнерну архітектуру, яка включає окремі сервіси для клієнтської частини, серверної логіки, бази даних та допоміжних сервісів. Такий підхід забезпечує модульність системи, спрощує масштабування та підвищує стабільність роботи застосунку.

Функціональність освітньої платформи передбачає роботу з різними типами файлів, включаючи аватари користувачів, вкладення до завдань, навчальні матеріали та медіаконтент. Для реалізації об'єктного сховища даних використано систему MinIO.

MinIO є високопродуктивним об'єктним сховищем, сумісним з API Amazon S3, що забезпечує просту інтеграцію та масштабованість. Його основними перевагами є:
– висока швидкість роботи з файлами;
– сумісність із S3-екосистемою;
– простота розгортання у контейнеризованому середовищі;
– підтримка масштабування та розподіленого зберігання.

У межах проєкту MinIO використовується як централізоване сховище для всіх користувацьких файлів. Взаємодія з ним реалізована через серверний сервіс, який забезпечує завантаження, отримання та видалення об'єктів. Для реалізації функціоналу електронних повідомлень у системі використовується інструмент MailHog, який призначений для тестування SMTP-відправлень у локальному середовищі. MailHog перехоплює всі вихідні електронні листи, не відправляючи їх реальним користувачам, що дозволяє безпечно тестувати функціонал розсилок. Основні можливості інструменту включають:
– запуск локального SMTP-сервера для перехоплення листів;
– веб-інтерфейс для перегляду та аналізу повідомлень;
– зберігання листів у пам'яті для зручного тестування;
– підтримку REST API для автоматизації перевірок;
– сумісність із сучасними мовами програмування та фреймворками.

У межах проєкту MailHog використовується для перевірки сценаріїв відправки листів під час реєстрації користувачів та запрошень до системи, що дозволяє уникнути помилкової відправки реальних повідомлень під час розробки.

Висновки

У роботі було створено інформаційно-тестувальну платформу для навчального закладу. Проведено дослідження та аналіз додатків, технологій, що є актуальними на даний момент у розробці. Розроблено серверну частину за допомогою популярного та потужного фреймворку NestJS та інших бібліотек. У проєкті реалізовано авторизацію та аутентифікацію користувача, встановлена чисельна кількість зв'язків у базі даних, розроблено функціонал обробки відповідей і оцінювання студентів, задокументовано інформацію про запити. Було практично досліджено ORM Prisma для роботи з даними у базі PostgreSQL.

Розроблено клієнтську частину платформи за допомогою бібліотеки React. Реалізовано інтерфейс користувача, зручну навігацію по сайту, реєстрацію користувачів, закладів, класів та предметів. Створено форму для тестування та публікації завдань.

Основні результати проведеної роботи полягають у наступному:

1. Досліджено вже існуючі навчальні проєкти.
2. Розглянуто середовища та засоби створення проєкту.
3. Реалізовано сервер.
4. Розроблено клієнтську сторону платформи.
5. Запропоновано систему інформування та тестування для учнів.
6. Досліджено бібліотеку готових компонентів та їхню стилізацію .
7. Здійснено тестування додатку.

Список використаної літератури:

1. Google Classroom [Електронний ресурс] : вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Google_Classroom. – Назва з екрану.
2. Єдина Школа [Електронний ресурс] : Єдина Школа. – Режим доступу: https://eschool-ua.com/#/home. – Назва з екрану.
3. OptimaSchool [Електронний ресурс] : OptimaSchool. – Режим доступу: https://optima.school/vstup/oplata. – Назва з екрану.
4. Greenfield J., Short K., Cook S., Kent S., Crupi J. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. — Indianapolis: Wiley Publishing, 2004. – 592 p.
5. 60 років базам даних / В.А. Резніченко // Проблеми програмування. – 2021. – № 3. – С. 40-71. – Бібліогр.: 211 назв. – укр.
6. Проектування баз даних [Електронний ресурс] : pidru4niki. – Режим доступу: https://pidru4niki.com/11570718/bankivska_sprava/proektuvannya_baz_danih#616. – Назва з екрану.
7. PostgreSQL [Електронний ресурс] : PostgreSQL. – Режим доступу: https://www.postgresql.org. – Назва з екрану.
8. pgAdmin [Електронний ресурс] : pgAdmin. – Режим доступу: https://www.pgadmin.org. – Назва з екрану.
9. Prisma [Електронний ресурс] : PostgreSQL. – Режим доступу: https://www.prisma.io/docs. – Назва з екрану.
10. Node.js IN ACTION: example-driven tutorial / [Mike Cantelon and others]. – [2-nd edition]. – Manning, 2017. – 36 p.
11. Why does Typescript exist [Електронний ресурс] : typescriptlang.org. – Режим доступу: https://www.typescriptlang.org/why-create-typescript. – Назва з екрану.
12. Про розробку додатків на Nest JS [Електронний ресурс] : foxminded.ua. – Режим доступу: https://foxminded.ua/nest-js. – Назва з екрану.
13. Swagger vs Postman | Top 10 Differences You Should Know [Електронний ресурс] : testsigma.com. – Режим доступу: https://testsigma.com/blog/swagger-vs-postman. – Назва з екрану.
14. JSON Web Token (JWT) [Електронний ресурс] : datatracker.ietf.org. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519. – Назва з екрану.
15. Про токени, JSON Web Tokens (JWT), аутентифікацію и авторизацію. Token-Based Authentication [Електронний ресурс] : https://gist.github.com. – Режим доступу: https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc. – Назва з екрану.
16. What is a JWT? Understanding JSON Web Tokens [Електронний ресурс] : supertokens.com. – Режим доступу: https://supertokens.com/blog/what-is-jwt. – Назва з екрану.

17. Node JS Send an Email [Електронний ресурс] : w3schools.com. – Режим доступу: https://www.w3schools.com/nodejs/nodejs_email.asp. – Назва з екрану.
18. Figma [Електронний ресурс] : вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Figma. – Назва з екрану.
19. Why Vite [Електронний ресурс] : vitejs.dev. – Режим доступу: https://vitejs.dev/guide/why.html. – Назва з екрану.
20. A beginner's guide to create SPA with React JS [Електронний ресурс] : dev.to. – Режим доступу: https://dev.to/hiteshtech/a-beginners-guide-to-create-spa-with-react-js-491c. – Назва з екрану.
21. Material UI - Overview [Електронний ресурс] : mui.com. – Режим доступу: https://mui.com/material-ui/getting-started. – Назва з екрану.
22. Getting Started with Redux [Електронний ресурс] : redux.js.org. – Режим доступу: https://redux.js.org/introduction/getting-started. – Назва з екрану.
23. What is Docker? [Електронний ресурс] : https://aws.amazon.com. – Режим доступу: https://aws.amazon.com/docker/?nc1=h_ls. – Назва з екрану.
24. Introduction to MinIO [Електронний ресурс] : baeldung.com. – Режим доступу: https://www.baeldung.com/minio. – Назва з екрану.
25. MailHog Tutorial [Електронний ресурс] : mailtrap.io. – Режим доступу: https://mailtrap.io/blog/mailhog-explained. – Назва з екрану.

References:

1. Google Classroom [Electronic resource]: Wikipedia. – Access mode: https://uk.wikipedia.org/wiki/Google_Classroom
2. "Unified School" [Electronic resource]: Unified School. – Access mode: https://eschool-ua.com/#/home
3. OptimaSchool [Electronic resource]: OptimaSchool. – Access mode: https://optima.school/vstup/oplata
4. Greenfield J., Short K., Cook S., Kent S., Crupi J. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. — Indianapolis: Wiley Publishing, 2004. – 592 p.
5. 60 years of databases / V.A. Reznichenko // Problems in Programming. – 2021. – No. 3. – P. 40–71. – Bibliography: 211 titles. – Ukrainian.
6. Database design [Electronic resource]: pidru4niki. – Access mode: https://pidru4niki.com/11570718/bankivska_sprava/proektuvannya_baz_danih#616
7. PostgreSQL [Electronic resource]: PostgreSQL. – Access mode: https://www.postgresql.org
8. pgAdmin [Electronic resource]: pgAdmin. – Access mode: https://www.pgadmin.org
9. Prisma [Electronic resource]: Prisma documentation. – Access mode: https://www.prisma.io/docs
10. Node.js in Action: Example-Driven Tutorial / Mike Cantelon et al. – 2nd ed. – Manning, 2017. – 36 p.
11. Why does TypeScript exist [Electronic resource]: typescriptlang.org. – Access mode: https://www.typescriptlang.org/why-create-typescript
12. About NestJS development [Electronic resource]: foxminded.ua. – Access mode: https://foxminded.ua/nest-js
13. Swagger vs Postman | Top 10 Differences You Should Know [Electronic resource]: testsigma.com. – Access mode: https://testsigma.com/blog/swagger-vs-postman
14. JSON Web Token (JWT) [Electronic resource]: datatracker.ietf.org. – Access mode: https://datatracker.ietf.org/doc/html/rfc7519
15. About tokens, JSON Web Tokens (JWT), authentication and authorization. Token-Based Authentication [Electronic resource]: gist.github.com. – Access mode: https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc
16. What is a JWT? Understanding JSON Web Tokens [Electronic resource]: supertokens.com. – Access mode: https://supertokens.com/blog/what-is-jwt
17. Node JS Send an Email [Electronic resource]: w3schools.com. – Access mode: https://www.w3schools.com/nodejs/nodejs_email.asp
18. Figma [Electronic resource]: Wikipedia. – Access mode: https://uk.wikipedia.org/wiki/Figma
19. Why Vite [Electronic resource]: vitejs.dev. – Access mode: https://vitejs.dev/guide/why.html
20. A beginner's guide to create SPA with React JS [Electronic resource]: dev.to. – Access mode: https://dev.to/hiteshtech/a-beginners-guide-to-create-spa-with-react-js-491c
21. Material UI – Overview [Electronic resource]: mui.com. – Access mode: https://mui.com/material-ui/getting-started
22. Getting Started with Redux [Electronic resource]: redux.js.org. – Access mode: https://redux.js.org/introduction/getting-started
23. What is Docker? [Electronic resource]: aws.amazon.com. – Access mode: https://aws.amazon.com/docker/?nc1=h_ls

24. Introduction to MinIO [Electronic resource]: baeldung.com. – Access mode: https://www.baeldung.com/minio
25. MailHog Tutorial [Electronic resource]: mailtrap.io. – Access mode: https://mailtrap.io/blog/mailhog-explained

VOYTSIKHOVSKA Lena,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DZYUBA Viktoria,
Candidate of Technical Sciences, Senior Lecturer in the Department of Applied Mathematics and Computer Science at Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT OF AN EDUCATIONAL TESTING WEB APPLICATION FOR AN EDUCATIONAL INSTITUTION

Summary. Introduction. *The fast pace of modern life is forcing us to adjust our plans and daily routines, compelling us to temporarily or permanently relocate, spend several hours in shelters, set priorities wisely, and conserve electricity. The whole world, including Ukraine, is currently undergoing a phase of digitalization so that anyone can handle basic tasks without leaving home. This includes shopping, filing paperwork, scheduling appointments with specialists, and similar activities. In this way, to some extent, we can ensure a successful balance between safety and comfort in people's lives.*

Education is also undergoing digitalization, because in the midst of such chaos, we must continue to learn and grow, regardless of location or the circumstances around us. To do this, everything we need must be in one place. Of course, technology cannot replace a teacher at this point, but supplementary materials, homework assignments, and grades–all hosted on a single website accessible to parents, students, and teachers–will simplify life and add flexibility to the educational process. This approach will save time and resources and be freely available to any educational institution or organization.

Purpose *of this article is to develop an informational and testing web application for an educational institution, equipped with basic and essential functionality, which can be further improved based on user feedback, professional development, and team expansion.*

Results. *The article examines the development of a modern web-based educational platform designed to support interaction between students, teachers, and school administrators. It describes the main stages of system development, including the analysis of existing educational platforms, identification of their advantages and limitations, and formulation of requirements for a unified and scalable learning management system.*

The proposed system integrates core educational processes into a single platform, enabling users to manage schools, classes, subjects, assignments, tests, and communication within one ecosystem. The main features of the system include user registration and authentication, role-based access control (student, teacher, administrator), creation and management of classes, assignment distribution, test creation and evaluation, file sharing, and interaction through posts and notifications. The system also supports invitation mechanisms via email and ensures secure access through token-based authentication.

Special attention is given to the architecture and implementation of the system, which is based on modern technologies such as PostgreSQL, Prisma ORM, Node.js, TypeScript, NestJS, React, Redux, Vite, and Material UI. These technologies provide high performance, modularity, and scalability of the system. The system also integrates additional services such as MinIO for file storage, Docker for containerization, and MailHog for testing email functionality. The client-side and server-side applications are designed as a unified full-stack solution ensuring consistent user experience across different devices.

The testing results confirm the correct operation of all system modules, including authentication, role management, data processing, file handling, and communication features. The developed platform is fully functional and suitable for practical use in educational institutions.

Conclusion. *The article presents a study on the design and implementation of a unified educational management system for schools. The development of such a system addresses the*

fragmentation of existing educational tools and improves the efficiency of communication and data management within educational institutions.

The proposed solution demonstrates the effectiveness of integrating various educational processes into a single digital platform, allowing users to manage academic activities in a structured and convenient way. This approach significantly simplifies interaction between students, teachers, and administrators while improving transparency and organization of learning processes.

The main outcomes of the work include:

- *analysis of existing educational platforms and identification of their limitations;*
- *formulation of functional requirements for a unified system;*
- *design and implementation of a full-stack web application using modern technologies;*
- *integration of authentication, role management, and educational workflows;*
- *testing and validation of system functionality and usability.*

The results of the study confirm that the developed system improves the efficiency of educational process management and can be further extended with additional features such as analytics, mobile applications, and external integrations in the future.

Keywords: *web application, information system, client-server architecture, database, PostgreSQL, Prisma, Node.js, TypeScript, NestJS, REST API, JWT, authentication, authorization, React, SPA, Redux, Vite, MUI, Figma, Docker, MinIO, MailHog, deployment, web development.*

Одержано редакцією 20.09.2024 р.
Прийнято до публікації 30.10.2024 р.

УДК 004.415.2:004.75

DOI 10.31651/2076-5886-2024-1-90-102

PACS 07.05.Tp, 89.20.Ff

ТКАЧЕНКО Олексій Олексійович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

ПІСКУН Олександр Варфоломійович
кандидат технічних наук, доцент, завідувач кафедри прикладної математики та інформатики, Черкаський національний університет ім. Б. Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ МУЛЬТИПЛЕЄРНОГО КЛАВІАТУРНОГО ТРЕНАЖЕРА З ВИКОРИСТАННЯМ REACT ТА WEBSOCKET-ТЕХНОЛОГІЙ

У роботі розглянуто підходи до проектування та розробки клієнтської частини мультиплеєрного веб-застосунку для тренування швидкості та точності клавіатурного

набору тексту. Проведено порівняльний аналіз наявних рішень у даній предметній галузі – TypingClub, Nitro Type та 10FastFingers – на підставі якого виявлено обмеження існуючих підходів і сформульовано вимоги до власного застосунку. Обґрунтовано вибір технологічного стеку: React.js із хуками та контекстом для побудови компонентної SPA-архітектури, TypeScript для статичної типізації, TailwindCSS для стилізації й підтримки тем оформлення, ActionCable на основі протоколу WebSocket (RFC 6455) для забезпечення низькозатримкового двостороннього зв'язку між учасниками гри в режимі реального часу, а також механізми авторизації на основі JWT (RFC 7519) та Google OAuth2. Описано архітектуру клієнтської частини застосунку: компонентну структуру, механізми маршрутизації (React Router), управління глобальним станом через React Context та обробку подій WebSocket-каналу. Детально розглянуто реалізацію центрального компонента ігрової кімнати, що підтримує два режими – онлайн-змагання та офлайн-практику, – таблиці лідерів із сортуванням і пошуком, а також мультиплеєрної сторінки з управлінням ігровими кімнатами. Порівняно застосовані технічні підходи з альтернативами. Встановлено, що реалізований застосунок поєднує можливості, відсутні в будь-якому з аналізованих аналогів.

Ключові слова: *клавіатурний тренажер, React.js, TypeScript, TailwindCSS, WebSocket, ActionCable, JWT, OAuth2, мультиплеєрний веб-застосунок, SPA, однобічний потік даних.*

Вступ

Зростання обсягів цифрової комунікації, поширення дистанційних форм роботи і навчання, а також збільшення частки текстоорієнтованих засобів взаємодії підвищують значущість навичок клавіатурного введення тексту. Ця компетенція є важливою не лише для ІТ-фахівців, а й для широкого кола користувачів: студентів, офісних працівників, журналістів, науковців та інших категорій. Дослідження в галузі ергономіки та методики навчання показують, що цілеспрямована практика із зворотним зв'язком є ефективнішою, ніж неструктуроване введення тексту.

Клавіатурні тренажери є перевіреним інструментом формування даної навички: вони надають зворотний зв'язок щодо швидкості та точності введення, дозволяють визначати слабкі місця і відслідковувати динаміку прогресу. Водночас більшість наявних рішень або орієнтовані виключно на індивідуальні тренування без соціальної складової, або пропонують мультиплеєрний режим без повноцінного аналітичного інструментарію і персоналізованого інтерфейсу.

З технічної точки зору мультиплеєрний тренажер реального часу є складнішою задачею, ніж індивідуальний. Потрібно забезпечити синхронну передачу стану між учасниками з мінімальною затримкою, коректно обробляти підключення та відключення гравців, підтримувати конкурентний доступ до ресурсів сервера і при цьому надавати відзивчивий інтерфейс. Такі вимоги формують специфічний технічний контекст, де вибір технологій для клієнтської частини безпосередньо впливає на якість кінцевого продукту.

Зазначені чинники визначають актуальність дослідження та практичну значущість розробки мультиплеєрного клавіатурного тренажера з повноцінною сучасною клієнтською частиною.

На сьогодні ринок освітніх веб-застосунків є одним із найактивніше зростаючих сегментів EdTech. Зростання попиту на дистанційне навчання та самовдосконалення в зручному темпі стимулює розробку нових інструментів, що поєднують навчальну ефективність із соціальною залученістю. Гейміфікація освітніх платформ – включення механік нагород, змагань, рейтингів і прогресивних досягнень – є доведеним інструментом підвищення мотивації та утримання користувачів. Саме тому поєднання гейміфікованого мультиплеєрного режиму з аналітичним інструментарієм у рамках єдиного сучасного веб-застосунку є актуальним науково-практичним завданням.

Обраний технологічний стек відповідає сучасному стану галузі фронтенд-розробки: React залишається однією з найширше застосовуваних бібліотек для

побудови SPA, TypeScript набув статусу стандарту де-факто для великих TypeScript-проектів, TailwindCSS стрімко набирає популярність як альтернатива CSS-in-JS рішенням, а WebSocket є промисловим стандартом для застосунків реального часу. Вибір зрілих, широко підтримуваних технологій з відкритим кодом забезпечує перспективу тривалої підтримки і розвитку системи.

Мета статті – розробити клієнтську частину мультиплеєрного клавіатурного тренажера з використанням технологічного стеку React.js, TypeScript, TailwindCSS та ActionCable/WebSocket, що забезпечує змагання в режимі реального часу, автентифікацію користувачів та відображення аналітики результатів.

Виклад основного матеріалу

1. Огляд існуючих рішень та аналіз літератури

Для визначення вимог до власного застосунку проведено аналіз трьох популярних веб-рішень у сфері тренування клавіатурного введення.

TypingClub [1] є одним із найбільш функціонально насичених тренажерів: він пропонує структуровану програму навчання з поетапним підвищенням складності, інтерактивні анімації та звуковий супровід. Перевага TypingClub – методологічна структурованість навчального контенту та зручний інтерфейс для початківців. Однак відсутність мультиплеєрного режиму є суттєвим обмеженням для користувачів, яким важлива змагальна складова, а також відсутня детальна аналітика за окремими ігровими сесіями.

Nitro Type [2] позиціонує себе як гейміфікований тренажер у форматі автомобільних перегонів із мультиплеєрним режимом. Змагальна механіка та ігрові елементи підвищують залученість і мотивацію. Разом із тим надмірний акцент на швидкості без рівного акценту на точності може негативно впливати на якість формованих навичок; детальна аналітика сесії не передбачена.

10FastFingers [3] – мінімалістичний, легкодоступний інструмент зі спрощеним інтерфейсом і підтримкою введення різними мовами; наявний режим змагань. Недоліком є відсутність структурованої навчальної програми, детальної аналітики прогресу та персоналізованого інтерфейсу.

Порівняльна характеристика аналізованих рішень подана у таблиці 1.

Аналіз показав: жодне з розглянутих рішень не поєднує повноцінний мультиплеєрний режим, детальну аналітику ігрових сесій у вигляді графіків, персоналізований інтерфейс зі зміною теми та сучасні механізми авторизації. Це підтверджує доцільність розробки власного рішення та формує чіткий орієнтир для проектування.

Таблиця 1

Порівняльна характеристика наявних клавіатурних тренажерів

Критерій	TypingClub	Nitro Type	10FastFingers
Мультиплеєр у реальному часі	Ні	Так	Частково
Структурована програма навчання	Так	Ні	Ні
Детальна аналітика сесії (графіки)	Ні	Ні	Ні
Персоналізація теми інтерфейсу	Ні	Частково	Ні
Таблиця лідерів із сортуванням	Ні	Так	Так
Авторизація через Google	Ні	Ні	Ні
Метрика «символи за секунду»	Ні	Ні	Так

З технічної точки зору, проблематика клієнтських застосунків реального часу широко розглядається у контексті вибору між технологіями: HTTP-опитування

(polling), Server-Sent Events та повнодуплексний WebSocket (RFC 6455 [14]). WebSocket забезпечує найнижчу затримку при двосторонньому обміні за рахунок єдиного постійного TCP-з'єднання. Питанням проектування React-застосунків, управління станом і обробки WebSocket-потоків приділяється значна увага у технічній документації та публікаціях, що підтверджує зрілість обраних технологій.

2. Постановка задачі та архітектура застосунку

На підставі виявлених недоліків аналогів сформульовано функціональні вимоги до розроблюваного застосунку:

1. Підтримка мультиплеєрного режиму з відображенням прогресу суперника в режимі реального часу.
2. Авторизація через власний обліковий запис (JWT) та через Google OAuth2.
3. Збереження та відображення аналітики ігрових сесій: швидкість у символах за секунду (sps) та точність (%).
4. Таблиця лідерів із можливістю пошуку за нікнеймом та двонапрямним сортуванням за ключовими метриками.
5. Перемикання між кількома кольоровими темами інтерфейсу.
6. Режим офлайн-практики без взаємодії з сервером (не зберігається у базі даних).

Клієнтська частина реалізована як односторінковий застосунок (SPA – Single Page Application) з компонентною архітектурою на базі React.js [4]. Підхід SPA дозволяє уникати повних перезавантажень сторінки при навігації, що критично для застосунків реального часу: підписки WebSocket і стан контексту зберігаються протягом усієї сесії без розривів.

Маршрутизація між сторінками здійснюється бібліотекою React Router [12]. Глобальний стан (дані авторизованого користувача, поточна тема) передається через React Context API [10], що усуває необхідність «прокидання» пропсів через численні рівні ієрархії.

Ключові маршрути застосунку:

– / – головна сторінка (Home): відображення останніх ігор, кнопки переходу до режимів;
– /gameRoom і /gameRoom/:room_id – ігрова кімната (офлайн і онлайн режими);
– /multiplayerRoom – перегляд і створення ігрових кімнат;
– /leaderboard – таблиця лідерів;
– /account – профіль користувача.

3. Технологічний стек клієнтської частини

3.1 React.js: компонентна архітектура та Virtual DOM

React.js [4] – бібліотека JavaScript для побудови користувацьких інтерфейсів, розроблена компанією Meta у 2013 році. Основна концепція – декомпозиція інтерфейсу на незалежні, повторно використовувані компоненти. Кожен компонент інкапсулює власну логіку стану та відображення, що сприяє модульності й полегшує тестування.

Ключовою особливістю React є механізм Virtual DOM – легковагової копії реального DOM-дерева. При зміні стану компонента React оновлює не реальний DOM безпосередньо, а спочатку вносить зміни у Virtual DOM, після чого алгоритм узгодження (reconciliation) визначає мінімальний набір операцій над реальним DOM. Це суттєво знижує навантаження при частих оновленнях інтерфейсу – властивість,

критична для ігрового тренажера, де стан (позиція каретки, прогрес гравців) змінюється при кожному натисканні клавіші.

Архітектура застосунку побудована виключно на функціональних компонентах із хуками:

- useState – управління локальним станом компонента (поточна позиція каретки, стан символів, швидкість тощо);
- useEffect – обробка побічних ефектів: HTTP-запити до сервера при завантаженні, підписки на WebSocket-канали, запуск/зупинка таймерів;
- useContext – доступ до глобального стану (UserContext, ThemeContext);
- useRef – зберігання посилань на DOM-елементи без ініціювання повторного рендерингу (наприклад, для фокусування поля введення).

Однобічний потік даних (one-way data flow) [5] забезпечує передбачуваність стану: дані передаються від батьківських компонентів до дочірніх виключно через пропси, а зворотний зв'язок здійснюється через callback-функції. Такий підхід спрощує відлагодження та унеможливлює несанкціоновану мутацію стану.

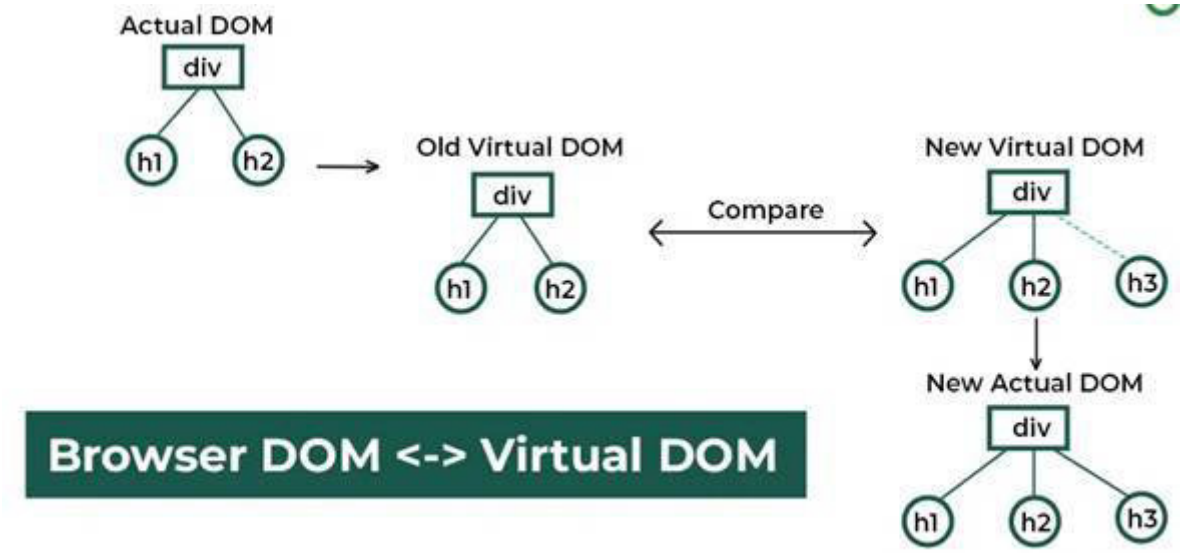


Рис. 1. Механізм Virtual DOM

3.2 TypeScript: статична типізація у динамічному середовищі

TypeScript [8] – надмножина JavaScript, розроблена компанією Microsoft, що додає систему статичних типів із виведенням типів і підтримку сучасних стандартів ECMAScript. Код TypeScript компілюється у стандартний JavaScript, сумісний з будь-яким сучасним браузером чи середовищем Node.js.

Статична типізація TypeScript дозволяє виявляти категорію помилок безпосередньо на етапі компіляції, до виконання програми. У контексті мультиплеєрного тренажера це особливо важливо в двох аспектах:

1. *Структури WebSocket-повідомлень*: визначення TypeScript-інтерфейсу для очікуваної структури повідомлення каналу RaceChannel гарантує, що звернення до поля зі швидкістю суперника не призведе до undefined за умови зміни формату серверної відповіді.

2. *Передача пропсів між компонентами*: визначення типів пропсів для кожного компонента, наприклад опису ігрової сесії для компонента SessionLineChart, усуває цілий клас помилок часу виконання, пов'язаних із неправильним форматом переданих даних.

Додаткові переваги у проекті: самодокументований код завдяки іменованим типам та інтерфейсам, розширена підтримка IDE (автодоповнення, рефакторинг, навігація за типами), підтримка всіх можливостей ECMAScript 6+ (класи, деструктуризація, генератори).

3.3 TailwindCSS: утилітарна стилізація та теми оформлення

TailwindCSS [9] – утилітарний CSS-фреймворк, що надає набір атомарних класів для стилізації безпосередньо у JSX-розмітці. На відміну від компонентних фреймворків на зразок Bootstrap, TailwindCSS не нав'язує готових компонентів – він надає примітиви (flex, grid, text-lg, bg-green-500 тощо), з яких розробник будує будь-який дизайн.

У розробленому застосунку TailwindCSS виконує дві ключові ролі:

1. *Базова стилізація компонентів***:** написання класів безпосередньо у JSX усуває необхідність підтримки окремих CSS-файлів і дозволяє бачити стилі поруч із розміткою.
2. *Підтримка п'яти кольорових тем* (зелена, червона, жовта, синя, чорна): завдяки механізму ThemeContext, що зберігає відповідний набір класів TailwindCSS, перемикання теми реалізовано динамічною підстановкою CSS-класів без написання окремих таблиць стилів для кожної теми та без перезавантаження сторінки.

Переваги підходу з TailwindCSS: висока швидкість розробки завдяки відмові від перемикання між файлами, адаптивний дизайн через префікси (sm:, md:, lg:), простота рефакторингу стилів.

3.4 WebSocket та ActionCable: двостороння комунікація в реальному часі

WebSocket [6] – протокол, стандартизований у RFC 6455 [14], що забезпечує повнодуплексний зв'язок між клієнтом і сервером через єдине постійне TCP-з'єднання. Після первинного HTTP-рукостискання (handshake) з'єднання переходить у стан WebSocket, після чого обидві сторони можуть надсилати фрейми даних у будь-який момент без нових запитів.

Порівняння доступних підходів до реалізації взаємодії в реальному часі подано у таблиці 2.

WebSocket обрано як оптимальний варіант: він забезпечує мінімальну затримку при двосторонньому обміні, що є критичним при відображенні прогресу суперника в режимі реального часу, і має широку підтримку у всіх сучасних браузерах.

ActionCable [7] – бібліотека на основі WebSocket, вбудована у фреймворк Ruby on Rails, що надає абстракцію каналів для організації підписок. Клієнтська частина підписується на канал RaceChannel; отримані повідомлення з полем швидкості суперника оновлюють відповідний стан компонента, що призводить до перерендерингу позиції «автомобіля» суперника на ігровому полі.

4. Реалізація механізмів авторизації

4.1 Управління глобальним станом через React Context

Механізм React Context [10] забезпечує передачу даних через дерево компонентів без «прокидання» пропсів через кожен проміжний рівень ієрархії. Фактично, це

глобальне сховище на рівні React-застосунку.

Таблиця 2

Порівняння технологій реального часу для клієнтських за стосунків

Технологія	**Механізм**	**Затримка**	**Напрямок**	**Накладні витрати**
HTTP short polling	Periodical GET	Висока	Клієнт → сервер	Великі (новий HTTP-запит кожного разу)
HTTP long polling	Утримуваний GET	Середня	Сервер → клієнт	Середні
Server-Sent Events	HTTP streaming	Середня	Лише сервер → клієнт	Низькі
WebSocket (RFC 6455)	TCP full-duplex	Низька	Двосторонній	Мінімальні після handshake

Визначено два контексти:

- UserContext – зберігає userId, userData (дані профілю) та стан авторизації. Після входу в систему всі компоненти, що споживають цей контекст (навігаційна панель, GameRoom, LeaderBoard, Account), отримують оновлені дані без додаткових запитів.
- ThemeContext – зберігає поточне значення теми та відповідний набір CSS-класів TailwindCSS. Перемикання теми через ThemeButton мгновенно оновлює стилі по всьому дереву компонентів.

4.2 JWT-авторизація

JSON Web Token (JWT) [11, 15] – стандарт (RFC 7519) компактного та самодостатнього способу передачі верифікованої інформації між сторонами у вигляді JSON-об'єкта з цифровим підписом. Токен складається з трьох частин: заголовка (алгоритм підпису), корисного навантаження (claims: userId, термін дії) та підпису.

Потік авторизації через JWT у застосунку:

1. Користувач вводить логін та пароль у компоненті LoginModal.
2. Клієнт надсилає POST-запит на сервер із обліковими даними.
3. Сервер перевіряє дані та повертає підписаний JWT.
4. Клієнт зберігає токен у localStorage і записує userId до UserContext.
5. При кожному HTTP-запиті або при встановленні WebSocket-з'єднання JWT включається у заголовок Authorization: Bearer <token>.
6. Сервер перевіряє підпис токена та термін його дії, після чого повертає захищений ресурс.

Перевага JWT перед традиційними серверними сесіями полягає у відсутності необхідності зберігати стан сесії на сервері: вся необхідна інформація знаходиться в самому токені, а сервер лише перевіряє цифровий підпис. Це спрощує горизонтальне масштабування серверної частини та дозволяє використовувати один токен для звернення до кількох мікросервісів або API-ендпоінтів.

Необхідно зауважити, що зберігання JWT у localStorage є прийнятним підходом для клієнтських SPA-застосунків при умові захисту від XSS-атак. У розробленому застосунку вхідні дані валідуються на стороні сервера, а зі сторони клієнта небезпечний HTML-вміст не вставляється у DOM без санітізації.

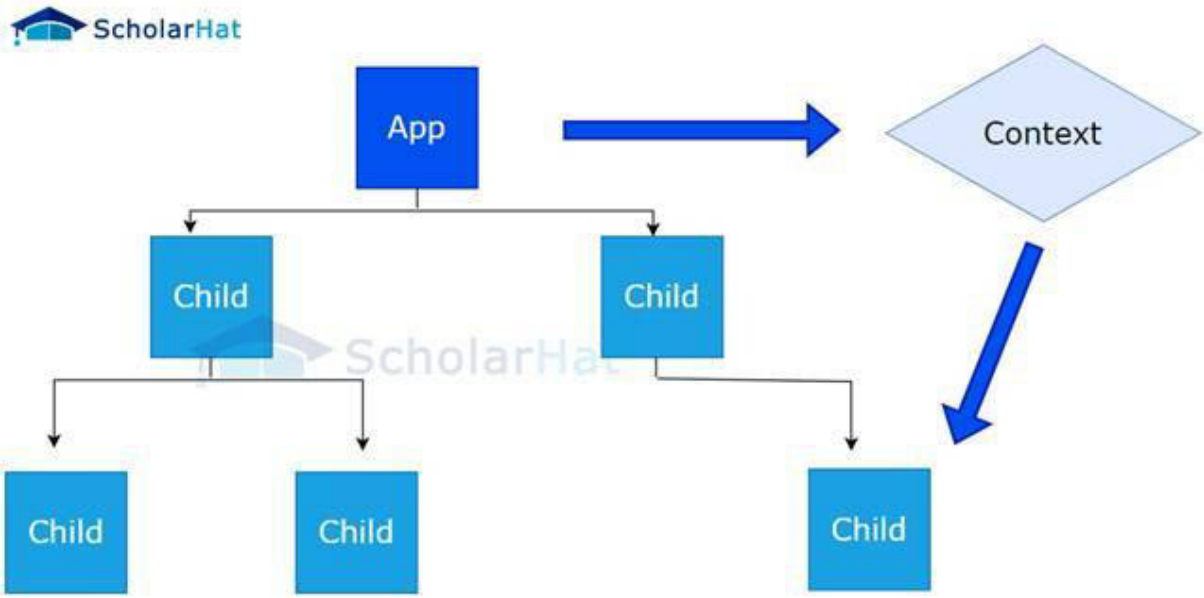


Рис. 4. Концепція контексту React (взято з сайту https://www.scholarhat.com/)

4.3 Авторизація через Google OAuth2

Альтернативний шлях авторизації реалізовано через Google OAuth2. Переваги: користувач не створює окремий пароль для застосунку, а сервер не зберігає паролі. Потік: клієнт перенаправляє користувача на сторінку Google → користувач підтверджує доступ → Google надає токен доступу → клієнт передає токен на сервер → сервер верифікує його через Google API, реєструє або авторизує користувача і повертає власний JWT.

5. Реалізація ігрової кімнати

Центральний компонент застосунку – GameRoom – відповідає за повний ігровий цикл. Компонент обслуговує два режими залежно від наявності параметра room_id у маршруті:

1. *Офлайн-режим* (/gameRoom): гравець практикується самостійно, результати відображаються після сесії, але не записуються у базу даних; WebSocket-підключення не використовується, що знижує навантаження на сервер.
2. *Онлайн-режим* (/gameRoom/:room_id): двосторонній WebSocket-зв'язок із сервером через RaceChannel; прогрес обох гравців синхронізується в реальному часі.

При завантаженні компонента в онлайн-режимі виконуються HTTP-запити:

1. /nick_names – отримання нікнеймів обох гравців кімнати;
2. /get_text – отримання тексту для поточної сесії;
3. /role – визначення ролі поточного користувача (хост або гість), що впливає на наявність кнопки «Start game».

Стан клавіатурного введення відслідковується посимвольно: для кожного символу підтримується статус correct / incorrect / pending. На основі цього обчислюється точність у реальному часі. Швидкість у символах за секунду (sps) обчислюється як відношення кількості правильно введених символів до часу від

початку гри та передається на сервер через sendMessage по WebSocket-каналу при кожній зміні.

Сервер трансльює отримане значення швидкості суперникові; клієнт відповідного гравця отримує повідомлення через підписку на RaceChannel і оновлює позицію «автомобіля» суперника. Ігровий процес таким чином синхронізується між двома клієнтами через сервер без прямого P2P-зв'язку.

Обробка помилок та граничних станів. При завантаженні онлайн-кімнати застосунок перевіряє приналежність user_id до поточної кімнати та відображає відповідне повідомлення: «Waiting for host to start…» якщо гравець не є хостом, «Start game» для хоста, «You are not allowed in this room» у разі спроби несанкціонованого доступу.

Анімація та візуалізація. Рух «автомобілів» відображається у форматі ASCII-art і оновлюється відповідно до поточної sps кожного гравця. Для плавних анімацій переходів між станами використовується Framer Motion [13]. Після завершення ігрової сесії компонент SessionLineChart відображає графік зміни швидкості введення впродовж сесії.

6. Реалізація таблиці лідерів та мультиплеєрної сторінки

6.1 Таблиця лідерів

Компонент LeaderBoard завантажує дані від ендпоінта /leaderboard методом GET і надає такі можливості взаємодії:

1. *Пошук* за нікнеймом: у хуку useMemo обчислюється масив filteredData – підмножина data, де поле nickname включає рядок із searchData. Результат оновлюється реактивно при кожній зміні введення.
2. *Сортування за точністю* (accuracy): двонапрямне сортування масиву за значенням відповідного поля; стан sortByAccuracy перемикається при кожному натисканні.
3. *Сортування за швидкістю* (sps): аналогічний механізм; при активації скидає стан sortByAccuracy.

Таблиця 3 порівнює метрики та можливості взаємодії в існуючих і розробленому рішеннях.

Таблиця 3

Метрики результативності та можливості таблиці лідерів

Метрика / функція	TypingClub	Nitro Type	10FastFingers	Розроблений застосунок
Кількість символів за секунду (sps)	Ні	Ні	Так	Так
Точність введення (%)	Так	Ні	Так	Так
Графік швидкості за сесію	Ні	Ні	Ні	Так
Таблиця лідерів	Ні	Так	Так	Так
Сортування в таблиці лідерів	Ні	Ні	Ні	Так
Пошук у таблиці лідерів	Ні	Ні	Ні	Так
Перегляд останніх ігор	Ні	Ні	Ні	Так

6.2 Мультиплеєрна сторінка

Компонент MultiplayerRoom відображає список активних ігрових кімнат, отриманих із сервера через ендпоінт /get_info_rooms. Підтримується сортування за

трьома критеріями:

1. *Наявність пароля* (password_status): кімнати з паролем на початку або в кінці списку.
2. *Кількість гравців* (players_count): заповнені або порожні кімнати – першими.
3. *Статус блокування* (game_lock_status): активні або заблоковані кімнати – першими.

Сортувальні стани є взаємовиключними: активація одного скидає інші. Пошук кімнат реалізовано за аналогією з LeaderBoard.

Для *створення нової кімнати* відображається модальне вікно, де користувач встановлює необов'язковий пароль. Після натискання «Create» виконується POST-запит на сервер; у відповідь повертається room_id, і клієнт перенаправляється до GameRoom з роллю хоста.

7. Аналіз результатів та наукова новизна

Розроблений застосунок реалізує декілька технічних підходів, сукупність яких відсутня в будь-якому з проаналізованих аналогів.

Поєднання однобічного потоку даних і двосторонньої WebSocket-комунікації. Стан ігрового поля є похідним від двох незалежних потоків: локального (клавіатурне введення) та зовнішнього (WebSocket-повідомлення від сервера). Розрізнення цих потоків у межах React-компонента вирішено через поділ стану: локальний стан управляється через useState, а ефекти підписки на WebSocket – через useEffect. Це забезпечує декларативне управління обома потоками в межах парадигми React без порушення принципів однобічного потоку даних.

Статична типізація для WebSocket-даних. Оголошення TypeScript-інтерфейсів для структур повідомлень каналу RaceChannel забезпечує перевірку кореляції між очікуваною і фактичною формою даних ще на етапі збірки. Це знижує ризик помилок виконання при зміні серверного API.

Динамічна тематизація через ThemeContext і TailwindCSS. Замість написання окремих CSS-файлів для кожної теми або використання CSS-змінних, підхід базується на динамічній підстановці наборів утилітарних класів TailwindCSS через контекст. Перемикання теми відбувається без перезавантаження сторінки та без втрати стану WebSocket-підписок.

Уніфікований компонент для двох режимів гри. Єдиний компонент GameRoom обслуговує офлайн-практику та онлайн-змагання, визначаючи режим через наявність параметра room_id у маршруті. Це зменшує дублювання коду і забезпечує консистентність ігрового інтерфейсу в обох режимах.

Порівняно з наявними рішеннями (TypingClub, Nitro Type, 10FastFingers) розроблений застосунок пропонує якісно відмінний набір можливостей: поєднання змагальної механіки, детальної аналітики сесій та персоналізованого інтерфейсу, заснованого на відкритому технологічному стеку зі стандартизованими протоколами (RFC 6455, RFC 7519).

Обмеження та напрями вдосконалення. Поточна реалізація підтримує лише двох гравців в одній кімнаті; розширення до груп потребуватиме перегляду архітектури каналу та структури даних. Зберігання JWT у localStorage є допустимим для навчального проекту, але у виробничому середовищі варто розглянути використання HttpOnly Cookies для підвищення стійкості до XSS. Адаптивний підбір текстів залежно від рівня підготовки користувача (наприклад, на основі частотності помилок по позиціях клавіатури) є перспективним напрямом, що зробить тренажер ефективнішим для цілеспрямованого вдосконалення навичок.

Загалом, результати роботи підтверджують, що сучасний стек React + TypeScript + TailwindCSS + WebSocket є достатнім і продуктивним рішенням для розробки інтерактивних освітніх застосунків реального часу в рамках кваліфікаційного проекту.

Висновки

У роботі розроблено клієнтську частину мультиплеєрного клавіатурного тренажера з використанням технологічного стеку React.js, TypeScript, TailwindCSS та ActionCable/WebSocket.

Проведений огляд наявних рішень (TypingClub, Nitro Type, 10FastFingers) виявив відсутність застосунку, що одночасно поєднує мультиплеєрний режим реального часу, детальну аналітику ігрових сесій, персоналізацію теми інтерфейсу та сучасні механізми авторизації.

Реалізовано такі ключові компоненти:

1) авторизація на основі JWT та Google OAuth2 із глобальним станом через React Context;
2) ігрова кімната (GameRoom) з підтримкою онлайн/офлайн режимів, посимвольним відстеженням точності та WebSocket-синхронізацією;
3) таблиця лідерів із пошуком і двонапрямним сортуванням;
4) мультиплеєрна сторінка з управлінням ігровими кімнатами;
5) механізм динамічної зміни теми через ThemeContext і TailwindCSS.

Обрані технологічні рішення забезпечують: мінімальну затримку відображення стану суперника (WebSocket, RFC 6455); надійність коду на етапі компіляції (TypeScript); модульність та повторне використання компонентів (React із хуками); гнучкість стилізації та підтримки тем (TailwindCSS).

Практичне значення розробки полягає у можливості використання застосунку як інструменту тренування навичок клавіатурного введення у навчальних закладах та для самостійного вдосконалення.

Перспективами подальшого розвитку є: впровадження алгоритмів адаптивного підбору тексту відповідно до поточного рівня підготовки користувача; розширення мультиплеєрного режиму до груп більше двох учасників; реалізація серверного рендерингу (SSR) для поліпшення продуктивності початкового завантаження; додавання інтернаціоналізації (i18n) для підтримки тренування введення різними мовами; впровадження Progressive Web App (PWA) для можливості офлайн-використання та встановлення на пристрій.

Розроблений застосунок демонструє доцільність застосування сучасного компонентного стеку (React + TypeScript + TailwindCSS) у поєднанні зі стандартизованими протоколами реального часу (WebSocket, RFC 6455) для побудови інтерактивних освітніх застосунків. Результати роботи можуть слугувати практичним орієнтиром для розробників аналогічних систем у сфері EdTech.

Список використаної літератури

1. TypingClub [Електронний ресурс]. – Режим доступу: https://www.typingclub.com (дата звернення: 20.05.2024).
2. Nitro Type [Електронний ресурс]. – Режим доступу: https://www.nitrotype.com (дата звернення: 20.05.2024).
3. 10FastFingers [Електронний ресурс]. – Режим доступу: https://10fastfingers.com (дата звернення: 20.05.2024).
4. React – A JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: https://reactjs.org (дата звернення: 29.04.2024).

5. React State and Lifecycle. One-way data flow [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/state-and-lifecycle.html (дата звернення: 29.04.2024).
6. WebSocket API [Електронний ресурс] / MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket (дата звернення: 19.05.2024).
7. How to use Action Cable with React and Rails [Електронний ресурс]. – Режим доступу: https://medium.com/swlh/how-to-use-action-cable-with-react-and-rails-3129a554c7d (дата звернення: 19.05.2024).
8. TypeScript – JavaScript with syntax for types [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org (дата звернення: 29.04.2024).
9. Tailwind CSS – A utility-first CSS framework [Електронний ресурс]. – Режим доступу: https://tailwindcss.com (дата звернення: 29.04.2024).
10. React Context API [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/context.html (дата звернення: 29.04.2024).
11. Introduction to JSON Web Tokens [Електронний ресурс]. – Режим доступу: https://jwt.io/introduction (дата звернення: 19.05.2024).
12. React Router [Електронний ресурс]. – Режим доступу: https://reactrouter.com (дата звернення: 29.04.2024).
13. Framer Motion – A production-ready motion library for React [Електронний ресурс]. – Режим доступу: https://www.framer.com/motion (дата звернення: 29.04.2024).
14. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455 [Електронний ресурс] / IETF. – 2011. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6455 (дата звернення: 19.05.2024).
15. Jones M. et al. JSON Web Token (JWT). RFC 7519 [Електронний ресурс] / IETF. – 2015. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 19.05.2024).

References

1. TypingClub [Electronic resource]. Access: https://www.typingclub.com (accessed: 20.05.2024).
2. Nitro Type [Electronic resource]. Access: https://www.nitrotype.com (accessed: 20.05.2024).
3. 10FastFingers [Electronic resource]. Access: https://10fastfingers.com (accessed: 20.05.2024).
4. React – A JavaScript library for building user interfaces [Electronic resource]. Access: https://reactjs.org (accessed: 29.04.2024).
5. React State and Lifecycle [Electronic resource]. Access: https://reactjs.org/docs/state-and-lifecycle.html (accessed: 29.04.2024).
6. WebSocket API [Electronic resource] / MDN Web Docs. Access: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket (accessed: 19.05.2024).
7. How to use Action Cable with React and Rails [Electronic resource]. Access: https://medium.com/swlh/how-to-use-action-cable-with-react-and-rails-3129a554c7d (accessed: 19.05.2024).
8. TypeScript [Electronic resource]. Access: https://www.typescriptlang.org (accessed: 29.04.2024).
9. Tailwind CSS [Electronic resource]. Access: https://tailwindcss.com (accessed: 29.04.2024).
10. React Context API [Electronic resource]. Access: https://reactjs.org/docs/context.html (accessed: 29.04.2024).
11. Introduction to JSON Web Tokens [Electronic resource]. Access: https://jwt.io/introduction (accessed: 19.05.2024).
12. React Router [Electronic resource]. Access: https://reactrouter.com (accessed: 29.04.2024).
13. Framer Motion [Electronic resource]. Access: https://www.framer.com/motion (accessed: 29.04.2024).
14. Fette I., Melnikov A. (2011) The WebSocket Protocol. RFC 6455. IETF. Access: https://datatracker.ietf.org/doc/html/rfc6455.
15. Jones M. et al. (2015) JSON Web Token (JWT). RFC 7519. IETF. Access: https://datatracker.ietf.org/doc/html/rfc7519.

TKACHENKO Oleksii,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Head of Department of Applied Mathematics and Informatics, Bohdan Khmelnytsky National University of Cherkasy

DEVELOPMENT OF THE CLIENT-SIDE PART OF A MULTIPLAYER KEYBOARD TRAINER USING REACT AND WEBSOCKET TECHNOLOGIES

Summary. Introduction. *The increasing volume of digital communication and the spread of remote work and learning enhance the importance of keyboard typing skills. This competence is essential not only for IT professionals but for a wide range of users: students, office workers, journalists, and researchers. Specialized typing trainers with feedback on speed and accuracy are a proven tool for developing this skill. However, most available solutions either focus exclusively on individual practice without a social component or provide multiplayer mode without comprehensive session analytics and personalized interface. From a technical perspective, a real-time multiplayer trainer presents challenges not present in individual trainers: synchronous state transmission with minimal latency, correct handling of player connections and disconnections, concurrent server resource access, and a responsive interface.*

Purpose. *The aim of this work is to develop the client-side part of a multiplayer keyboard trainer using the technology stack of React.js, TypeScript, TailwindCSS, and ActionCable/WebSocket, enabling real-time competitions, user authentication, and session result analytics.*

Results. *A comparative analysis of three popular keyboard trainer web applications – TypingClub, Nitro Type, and 10FastFingers – was conducted. The analysis revealed that none of these solutions combines real-time multiplayer, detailed session analytics in graphical form, personalized interface with theme switching, and modern authentication mechanisms. Based on this, the requirements for the developed application were defined.*

The client-side is implemented as a Single Page Application (SPA) using React.js with hooks and the Context API for state management. TypeScript provides compile-time type safety, which is especially important for WebSocket message structures. TailwindCSS enables dynamic theme switching across five color schemes without separate CSS files. ActionCable over the WebSocket protocol (RFC 6455) ensures low-latency bidirectional communication between game participants. Two authentication flows are implemented: JWT-based (RFC 7519) and Google OAuth2. The GameRoom component supports both offline practice and online multiplayer modes, determined by the presence of the room_id route parameter; character-by-character accuracy tracking and speed synchronization via WebSocket are implemented. The LeaderBoard component provides search and bidirectional sorting by accuracy and speed. The MultiplayerRoom component enables browsing and creating game rooms with sorting by password status, player count, and lock status.

Conclusion. *The developed application combines real-time multiplayer functionality, detailed session analytics, flexible interface personalization, and standard-based authentication mechanisms – a combination absent in any of the analyzed existing solutions. The technology stack ensures code reliability at compile time (TypeScript), minimal latency in multiplayer synchronization (WebSocket, RFC 6455), component reusability and predictable state management (React with one-way data flow), and styling flexibility (TailwindCSS). Directions for further development include adaptive text selection algorithms based on user skill level, support for group competitions with more than two participants, and server-side rendering for improved initial load performance.*

Keywords: *keyboard trainer, React.js, TypeScript, TailwindCSS, WebSocket, ActionCable, JWT, OAuth2, multiplayer web application, SPA, one-way data flow.*

Одержано редакцією 05.11.2024 р.
Прийнято до публікації 11.12.2024 р.

ЗМІСТ

СЕКЦІЯ «ПРИКЛАДНА МАТЕМАТИКА»

СЕКЦІЯ «ІНФОРМАТИКА»

CONTENTS

APPLIED MATHEMATICS SECTION

INFORMATICS SECTION

ВІСНИК ЧЕРКАСЬКОГО УНИВЕРСИТЕТУ

Серія Прикладна математика. Інформатика
№1.2024

Відповідальний за випуск
Пасічний М.О.

Відповідальний секретар
Сердюк О.А.

Комп'ютерне верстання
Сердюк О.А.

Підписано до друку 20.12.2024.
Формат 60х84/в. Папір офсет. Друк офсет. Гарнітура Times.
Ум. друк. арк. 10,1. Наклад 100 прим.