УДК 004.01

Хуторна Д. В., здобувачка 2 курсу
спеціальності 122 Комп'ютерні науки,
Фриз І. В., канд. фіз.-мат. наук, старший
викладач кафедри інформаційних технологій

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ХЕШ-ТАБЛИЦЬ ТА ЗБАЛАНСОВАНИХ ДЕРЕВ ПОШУКУ В АЛГОРИТМІЗАЦІЇ

Донецький національний університет імені Василя Стуса, м. Вінниця

Асоціативні масиви, або словники, є базовим компонентом у сучасному програмуванні. Вони дають змогу зберігати пари «ключ-значення» та виконувати ефективний доступ до значення за заданим ключем. Найпоширенішими структурами даних для реалізації словників є хеш-таблиці та збалансовані дерева пошуку. Ці структури мають спільне функціональне призначення, однак різняться як за внутрішніми механізмами, так і за характеристиками ефективності в різних умовах. Метою цього дослідження є порівняльний аналіз переваг та недоліків хеш-таблиць і збалансованих дерев пошуку з урахуванням практичних аспектів їх застосування.

Хеш-таблиця виконує відображення ключа у певну позицію масиву за допомогою хеш-функції. За умови рівномірного розподілу хешів доступ до елементу в середньому здійснюється за сталий час $O(1)$. Саме очікувана (середня) швидкодія є головною перевагою цієї структури [1]. Однак у разі великої кількості колізій використання неякісної хеш-функції або надмірного коефіцієнта заповнення час доступу може зрости до $O(n)$. Попри високу практичну ефективність, хеш-таблиці не зберігають ключі впорядковано та не дають змоги виконувати діапазонні запити або здійснювати послідовну ітерацію у відсортованому вигляді без додаткових структур [2].

Збалансовані дерева пошуку (зокрема AVL-дерева та червоно-чорні дерева) працюють за іншим принципом: кожна операція вставки або видалення супроводжується ребалансуванням, щоб висота дерева залишалася логарифмічною відносно кількості вузлів. Завдяки цьому час виконання пошуку, вставки та видалення гарантується як $O(\log n)$ у найгіршому випадку. Дерева підтримують впорядкованість ключів, що забезпечує можливість виконання діапазонних запитів, пошуку мінімального та максимального значення, послідовної ітерації за зростанням ключів. Це надає перевагу вибору збалансованих дерев у задачах, де важлива не тільки швидкість доступу, а й структура даних.

Вибір між хеш-таблицею та збалансованим деревом тісно пов'язаний з компромісом між середньою швидкодією та гарантованою у найгіршому випадку ефективністю. Хеш-таблиці значно швидші в типовому сценарії, проте не забезпечують стабільності за несприятливого розподілу ключів. Натомість дерева стабільні за продуктивністю, але потребують більше часу на виконання базових операцій, порівняно з ідеальною поведінкою хеш-таблиці [3].

Ще однією важливою відмінністю є відношення цих структур до пам'яті. Хеш-таблиці зазвичай резервують більший масив, ніж поточна кількість еле-

ментів, що дає змогу зменшити колізії, але збільшує споживання пам'яті. Дерева працюють ефективніше з погляду використання пам'яті, оскільки щоразу збільшуються лише на один вузол без необхідності розширення масиву.

У практичних застосуваннях хеш-таблиці найчастіше використовуються у мовах програмування як реалізація словників і множин (Python dict, Java HashMap, C++ unordered_map). Вони ефективні для кешування, пошуку у таблицях символів та обробки запитів з унікальними ключами, де впорядкованість не потрібна [1]. Збалансовані дерева, навпаки, широко застосовуються у базах даних, файлових системах та індексах, де важливо підтримувати впорядковані ключі. Прикладом є B-дерева і B+-дерева у СУБД, як-от MySQL і PostgreSQL [4]. Також дерева використовують у системах, що обробляють діапазонні запити, нечіткі пошуки, лексикографічну навігацію або завдання зі збереженням порядку.

Підсумовуючи, можна зазначити: вибір між хеш-таблицею та збалансованим деревом пошуку залежить від характеру задачі. Якщо необхідна висока швидкість доступу за ключем і немає вимог до впорядкування, оптимальним вибором є хеш-таблиця. Якщо ж важливо зберігати порядок елементів або гарантувати стабільну продуктивність незалежно від вхідних даних, треба обирати збалансоване дерево. Успішне застосування тієї чи іншої структури залежить від врахування особливостей предметної області, природи вхідних даних та пріоритетів ефективності.

Список використаних джерел

1. Sedgewick R., Wayne K. Algorithms. 4th ed. Boston: Addison-Wesley, 2011. 955 p.
2. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 3rd ed. Cambridge: MIT Press, 2009. 1312 p.
3. Goodrich M., Tamassia R., Goldwasser M. Data Structures and Algorithms in Java. 6th ed. Wiley, 2014.
4. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill, 2020.

УДК 004.925.8

Родюк А. І., здобувачка 3 курсу спеціальності 111 Математика,
Комаров В. Ф., канд. техн. наук, старший викладач кафедри прикладної математики та кібербезпеки

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ПОБУДОВИ ФРАКТАЛЬНИХ СТРУКТУР

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Фрактальна геометрія є ключовим інструментом моделювання нелінійних процесів. Актуальність теми підтверджується дослідженнями, де розглядаються фрактальні генеративні моделі [1] та оптимізація ітераційних процесів [2]. Проте більшість праць зосереджена на обчисленні розмірностей [3], залишаючи поза увагою комплексне порівняння алгоритмів побудови візуальних образів.