Висновки. Розроблений гібридний алгоритм дає змогу нівелювати обмеження класичних методів рядкового зіставлення. Алгоритм Рабіна–Карпа забезпечує лінійну складність пошуку цитувань, тоді як метод шинглів гарантує стійкість системи до лексичних модифікацій.

Список використаних джерел

1. Introduction to algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 4th ed. Cambridge, MA: The MIT Press, 2022. 1312 p. URL: https://books.google. com.ua/books?id=NLngYyWFl_YC&printsec=copyright&hl=uk#v=onepage&q&f=false (дата звернення: 27.11.2025).

2. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. 3rd ed. Cambridge: Cambridge University Press, 2020. P. 75–88. URL: http://www.mmds.org/ (дата звернення: 27.11.2025).

3. Meuschke N., Gipp B. State-of-the-art in detection of academic plagiarism. *Frontiers in Computer Science*. 2024. URL: https://www.frontiersin.org (дата звернення: 27.11.2025).

УДК 004.422.5

Вершигора Р. О., здобувач
2 курсу спеціальності 122 Комп'ютерні науки,
Потапова Н. А., канд. екон. наук, доцент,
доцент кафедри інформаційних технологій

ОПТИМІЗАЦІЯ ОБРОБКИ ДАНИХ ЗА ДОПОМОГОЮ ХЕШ-СТРУКТУР У C#

Донецький національний університет імені Василя Стуса, м. Вінниця

Оптимізація обробки даних за допомогою хеш-структур у C# набуває все більшої значущості у сучасному програмуванні, враховуючи зростаючі обсяги інформації та потребу у швидкому доступі до неї. Використання хеш-структур дає змогу зменшити час пошуку, вставки та видалення елементів, що суттєво підвищує продуктивність програмних систем [1]. Ця тема є актуальною для дослідників та розробників, які прагнуть створити масштабовані та ефективні додатки.

Хеш-структури – це типи даних, які реалізують швидкий доступ до об'єктів за допомогою обчислення індексів через хеш-функції. Основний елемент – пара ключ-значення, де ключ перетворюється у числовий індекс. У C# широко використовуються три основні хеш-структури: Hashtable, Dictionary і HashSet. Hashtable підтримує зберігання необмеженого обсягу пар без суворої типізації, Dictionary є більш сучасною типізованою колекцією, що підвищує швидкість і безпеку, а HashSet [2] зберігає унікальні елементи, що зручно для фільтрації та перевірки належності даних.

Основний принцип роботи хеш-структур полягає у конвертації ключа в хеш-код за допомогою хеш-функції, який потім відображається у позицію зберігання

в масиві. Завдяки цьому доступ до елементу має амортизований час O(1), що значно швидше за традиційний послідовний пошук [3]. Хеш-функція має забезпечувати мінімальні колізії – ситуації, коли різні ключі мають однаковий індекс, для підвищення ефективності.

Практичне застосування хеш-структур у C# охоплює кешування даних, швидке визначення унікальності елементів, індексацію у базах даних і реалізацію пошукових алгоритмів [1], [3]. У базах даних вони знижують часові витрати на вибірку даних, у кешах дають змогу швидко перевіряти наявність інформації без додаткових запитів. В алгоритмах пошуку й сортування хеш-структури допомагають уникнути дублювань і оптимізують логіку доступу до елементів.

Однак хеш-структури не позбавлені проблем. Колізії, що виникають через співпадіння хеш-кодів, негативно впливають на продуктивність. Для їх усунення застосовують методи відкритого та закритого хешування, як-от лінійне зонування, квадратичне зонування та використання ланцюжків [2]. Управління пам’яттю є ще однією важливою проблемою, особливо у великих проєктах та багатопотокових середовищах. Для таких випадків в C# рекомендовано використовувати потокобезпечні структури, зокрема ConcurrentDictionary, що оптимізує доступ і синхронізацію у багатопотоковому режимі [4].

Підсумовуючи, хеш-структури є потужним інструментом для оптимізації обробки даних у C#. Вони дають змогу час доступу до інформації, ефективно працюють з великими обсягами даних і забезпечують безпечну роботу у багатопотокових додатках. Вивчення їх особливостей і методів підвищення продуктивності є важливим для розробників, що прагнуть створювати швидкісні та надійні програмні продукти.

Список використаних джерел

1. Курс «Алгоритми та структури даних на C#». URL: https://www.youtube.com/watch?v=DvIjpscPIoI (дата звернення 25.11.2025).

2. Грудзинський Ю. Є. Алгоритми та структури даних: навч. посіб. Київ: КПІ ім. І. Сікорського, 2023. URL: https://ela.kpi.ua/server/api/core/bitstreams/0db974f9-16fa-459c-9f19-fab0021222ed/content (дата звернення 25.11.2025).

3. Wengrow J. A Common-Sense Guide to Data Structures and Algorithms. 2nd edition: Level Up Your Core Programming Skills. Pragmatic Bookshelf, 2020. 508 p.

4. Microsoft Docs. Compute hash values in C#. URL: https://learn.microsoft.com/ru-ru/troubleshoot/developer/visualstudio/csharp/language-compilers/compute-hash-values (дата звернення 25.11.2025).