УДК 004.415:004.62

ПРОГРАМНА ПЛАТФОРМА БЕНЧМАРКИНГУ АЛГОРИТМІВ СОРТУВАННЯ З ПРАКТИЧНИМИ РЕКОМЕНДАЦІЯМИ

Анастюк К.В.
Національний університет «Одеська політехніка»
Україна, Одеса
kanastiuk@gmail.com

Анотація. *У роботі розглянуто підхід до розробки програмної платформи для комплексного бенчмаркінгу алгоритмів сортування. Запропоновано модель системи, що включає модулі генерації тестових даних, виконання тестів, аналізу результатів та формування практичних рекомендацій. Платформа реалізована з використанням сучасного стеку технологій (FastAPI, React, Pandas) та забезпечує порівняльний аналіз алгоритмів на різних мовах програмування. Експериментальна перевірка підтвердила ефективність системи для вибору оптимальних алгоритмів сортування відповідно до специфіки даних та обмежень ресурсів.*

Ключові слова: *алгоритми сортування; бенчмаркінг; програмна платформа; порівняльний аналіз; багатомовність; оптимізація продуктивності.*

Вступна частина. Сортування залишається однією з найбільш поширених та ресурсомістких обчислювальних операцій у сучасному програмному забезпеченні. Незважаючи на наявність вбудованих оптимізованих функцій сортування в сучасних мовах програмування, вибір оптимального алгоритму для конкретних умов залишається складною задачею [1]. Існуючі дослідження продуктивності алгоритмів сортування мають значні розбіжності через різні тестові середовища, методики вимірювання та набори даних [2].

Основним недоліком сучасних підходів є відсутність уніфікованої системи для порівняльного аналізу алгоритмів на різних мовах програмування з урахуванням різноманітних типів даних та їх станів. Більшість існуючих рішень орієнтовані на окремі аспекти тестування та не надають структурованих рекомендацій для практичного застосування.

Метою дослідження є розробка програмної платформи для комплексного бенчмаркінгу алгоритмів сортування, яка забезпечує об'єктивне порівняння продуктивності різних алгоритмів на популярних мовах програмування та формування практичних рекомендацій щодо їх вибору.

Основна частина. У ході дослідження було розроблено концептуальну модель системи бенчмаркінгу, що ґрунтується на принципі комплексної оцінки алгоритмів через аналіз їхньої поведінки в різних умовах. Модель включає чотири основні компоненти:

1. Модуль генерації тестових даних реалізує методи створення різноманітних наборів даних. Для цілих чисел використовуються алгоритми генерації випадкових значень у діапазоні ±1,000,000, відсортованих послідовностей, майже відсортованих масивів (з 5% випадкових перестановок) та наборів з великою кількістю дублікатів. Для рядків реалізовано випадкові рядки довжиною від 5 до 50 символів, що містять символи різних регістрів та цифри, відсортовані набори рядків, що імітують впорядковані словникові дані

Для складних об'єктів генеруються структури даних із полями різних типів (ідентифікатор, рядок, числове значення, булевий прапорець), що імітують реальні об'єкти предметної області. Кожен тип даних генерується у декількох варіантах для забезпечення статистичної значущості результатів.

2. Модуль виконання тестів забезпечує інтеграцію з п'ятьма мовами програмування (Python, Java, C++, JavaScript, C#) через уніфіковані інтерфейси. Для кожної мови реалізовано набір алгоритмів сортування: швидке сортування (Quick Sort), сортування злиттям (Merge Sort), пірамідальне сортування (Heap Sort), сортування Шелла (Shell Sort), гібридний алгоритм Timsort та спеціалізовані алгоритми для конкретних типів даних (Counting Sort, Radix Sort) [3]. Важливим аспектом є включення до тестування стандартних методів сортування кожної мови програмування: std::sort у C++, Arrays.sort() у Java, sorted() у Python, вбудованих методів сортування у C# та JavaScript. Вимірювання продуктивності проводиться з точністю до мілісекунди з використанням системного таймера. Моніторинг споживання пам'яті реалізовано через бібліотеку psutil, що дозволяє відстежувати використання оперативної пам'яті під час виконання кожного алгоритму.

3. Система аналізу результатів використовує статистичні методи для обробки отриманих даних. Для кожного алгоритму розраховуються середній час виконання, стандартне відхилення, мінімальне та максимальне значення на основі декількох вимірювань для кожної комбінації параметрів. Аналіз масштабованості проводиться шляхом побудови графіків залежності часу виконання від розміру вхідних даних для кожного алгоритму. Визначається асимптотична поведінка алгоритмів на великих наборах даних (до 10,000,000 елементів) та їх ефективність на малих наборах (1,000-10,000 елементів).

4. Механізм формування рекомендацій ґрунтується на класифікації алгоритмів за критеріями продуктивності та стабільності [4]. Система визначає "профіль ефективності" для кожного алгоритму, що включає оптимальні умови його застосування: тип даних, розмір набору, стан даних (відсортовані, випадкові, з дублікатами), обмеження пам'яті.

Архітектура платформи реалізована на основі багаторівневої модульної структури. Веб-інтерфейс розроблений з використанням React та Bootstrap 5, що забезпечує адаптивний дизайн та інтуїтивну навігацію. Бекенд-система побудована на FastAPI, що забезпечує асинхронну обробку запитів та високу продуктивність при паралельному виконанні тестів. Для зберігання та аналізу даних використовуються Pandas та NumPy, що дозволяє ефективно працювати з великими обсягами результатів тестування. Система забезпечує ізоляцію процесів для кожного тестового запуску, що запобігає взаємному впливу та забезпечує стабільність вимірювань.

Експериментальна перевірка системи проводилась на наборах даних розміром від 1,000 до 1,000,000 елементів. Результати виявили помітну залежність ефективності як від типу даних та їх стану, так і від мови програмування:

На малих випадкових датасетах (1,000-10,000 елементів) стандартні методи сортування демонстрували найкращу продуктивність для більшості типів даних, завдяки високій оптимізації на рівні компілятора та використанню гібридних підходів.

Продуктивність ідентичних алгоритмів відрізнялась між мовами. Наприклад, Quick Sort на C++ був помітно швидший за аналогічну реалізацію на Python через різницю в інтерпретації та компіляції.

Для майже відсортованих даних Timsort перевершував стандартні методи на приблизно 15-20% у мовах, де він не використовується як основний алгоритм. Для даних з великою кількістю дублікатів Counting Sort демонстрував лінійну залежність часу виконання від розміру даних, що робить його оптимальним вибором для таких сценаріїв.

Таким чином, експериментальні дані підтверджують, що стандартні методи сортування є оптимальним вибором для більшості практичних завдань, особливо на малих та середніх наборах даних. Однак для специфічних сценаріїв та екстремальних умов спеціалізовані алгоритми можуть демонструвати значну перевагу у продуктивності.

Висновки. У роботі розроблено програмну платформу для комплексного бенчмаркінгу алгоритмів сортування, яка дозволяє проводити об'єктивне порівняння продуктивності різних алгоритмів на популярних мовах програмування. Експериментально підтверджено ефективність системи для вибору оптимальних алгоритмів сортування відповідно до специфіки даних та обмежень ресурсів.

Запропонований підхід відрізняється від існуючих рішень уніфікованою методологією тестування, підтримкою багатомовності та комплексним аналізом результатів з формуванням практичних рекомендацій. Платформа може бути використана розробниками для прийняття обґрунтованих рішень щодо вибору алгоритмів сортування, а також у навчальному процесі для демонстрації характеристик різних алгоритмів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). The MIT Press.

[2] Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley Professional.

[3] Knuth, D. E. (1998). *The Art of Computer Programming: Volume 3: Sorting and Searching* (2nd ed.). Addison-Wesley Professional.

[4] Auger, A., & Doerr, B. (2011). *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing.

Anastiuk K.V.

Software Platform for Sorting Algorithms Benchmarking with Practical Recommendations

Abstract. *The paper presents an approach to developing a software platform for comprehensive benchmarking of sorting algorithms. A system model is proposed that includes modules for test data generation, test execution, results analysis, and practical recommendations formation. The platform is implemented using a modern technology stack (FastAPI, React, Pandas) and provides comparative analysis of algorithms across different programming languages. Experimental verification confirmed the system's effectiveness for selecting optimal sorting algorithms according to data specifics and resource constraints.*

Keywords*: sorting algorithms; benchmarking; software platform; comparative analysis; multilingual support; performance optimization.*