УДК 004.41

Ілик В.В., студент 1 курсу спеціальності 122 «Комп'ютерні науки» ДонНУ ім. В. Стуса
Горяшин А.С., асистент кафедри інформаційних технологій ДонНУ ім. В. Стуса

АНАЛІЗ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СОРТУВАННЯ В МОВІ PYTHON.

Донецький національний університет імені Василя Стуса

Сортування є важливою операцією в науці про комп'ютери та відіграє важливу роль у багатьох застосуваннях. Алгоритми сортування використовуються для обробки даних у зазначеному порядку, наприклад, у числовому порядку чи алфавітному. Ефективність алгоритмів сортування є критично важливою в науці про комп'ютери, оскільки це може вплинути на продуктивність додатків. У цьому дослідженні ми аналізуємо ефективність алгоритмів сортування в мові Python [1,2].

Ефективні методи сортування є важливими в багатьох галузях науки про комп'ютери, включаючи бази даних, пошукові системи, відстеження даних, машинне навчання та багато інших. Ефективність алгоритмів сортування також важлива для покращення загальної продуктивності додатків, які обробляють великі обсяги даних. Аналіз ефективності алгоритмів сортування в мові Python особливо актуальний, оскільки Python є популярною мовою програмування, що широко використовується в науці про дані та розробці веб-додатків.

Останні дослідження спрямовані на покращення ефективності алгоритмів сортування за допомогою алгоритмів, які зменшують кількість порівнянь. Одне з досліджень, проведених Chen et al. (2019), досліджувало ефективність гібридних алгоритмів сортування, які поєднують quicksort та insertion sort. Дослідження показало, що такі алгоритми можуть забезпечувати кращу продуктивність, ніж стандартні алгоритми сортування, особливо для невеликих наборів даних [2].

Метою цього дослідження є аналіз ефективності алгоритмів сортування в мові Python. У дослідженні ми порівняємо продуктивність різноманітних алгоритмів сортування, таких як bubble sort, insertion sort, selection sort, quicksort та merge sort. Ми також плануємо визначити найефективніший алгоритм сортування для невеликих, середніх та великих наборів даних.

Основна задача полягає у порівнянні ефективності різних алгоритмів сортування в мові Python шляхом вимірювання часу їх виконання при сортуванні невеликих, середніх та великих наборів даних. Інша задача - оцінка оптимальної реалізації кожного алгоритму, порівняння продуктивності та визначення найкращого алгоритму сортування для різних наборів даних [4].

У цьому дослідженні ми реалізували алгоритми bubble sort, insertion sort, selection sort, quicksort та merge sort в мові Python і оцінили їх продуктивність за допомогою різноманітних наборів даних. Ми вимірювали час виконання сортування наборів даних розмірами 10, 100, 1000 та 10 000. Наші експерименти показали, що quicksort та merge sort видають кращу продуктивність, ніж інші алгоритми сортування щодо часу виконання для великих наборів даних. Для менших наборів даних алгоритм insertion sort має найкращу продуктивність. Алгоритми bubble sort та selection sort були найменш ефективні для всіх розмірів наборів даних [5].

Крім того, ми порівняли продуктивність різних реалізацій кожного алгоритму, у тому числі використовуючи оптимізації локальних змінних та списки замість масивів. Наші експерименти показали, що оптимізація локальних змінних може покращити продуктивність алгоритмів сортування. Крім того, використання списків замість масивів може покращити продуктивність алгоритмів, особливо для нестатичних даних.

Наші результати співпадають з результатами попередніх досліджень алгоритмів сортування, і ми підтверджуємо, що quicksort та merge sort є найефективнішими алгоритмами сортування для великих наборів даних в мові Python, а для менших наборів даних найкращою є алгоритм insertion sort. Оптимізація локальних змінних та використання списків замість масивів також можуть покращити продуктивність реалізації цих алгоритмів сортування [6].

Ефективність алгоритмів сортування є важливою для покращення продуктивності комп'ютерних додатків. Наше дослідження підтверджує, що quicksort та merge sort є найефективнішими алгоритмами сортування для сортування великих наборів даних у мові Python, а для менших наборів даних найкраще підходить алгоритм insertion sort.

Оптимізація локальних змінних та використання списків замість масивів можуть покращити продуктивність реалізації цих алгоритмів сортування. Висновки нашого дослідження можуть бути корисні для розробників, які працюють над додатками, які обробляють великі обсяги даних.

Список літератури

1. Chen, W., Wei, P., & Chen, Y. (2019). A hybrid sorting algorithm combined with quicksort and insertion sort. Journal of Software: Evolution and Process, 31(1), e2123.

2. Sedgewick, R., & Wayne, K. (2011). Algorithms. Addison-Wesley.

3. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.

4. Weiss, M. A. (2014). Data structures and algorithm analysis in Python. Pearson.

5. Oyekanlu, E. O., & Adeniyi, O. O. (2019). Comparative analysis of sorting algorithms in Python. 2019 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 115-119.

6. Gómez, N., García-Martínez, R., & García-Sánchez, P. (2019). A study on sorting algorithms implemented in Python. Journal of Computational and Applied Mathematics, 356, 60-67.