УДК 004.4(2)-004.42

Дорофєєв Є. О., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,
Вєтров О. С., старший викладач кафедри прикладної математики та кібербезпеки

АЛГОРИМ ЗЛИТТЯ В МОВІ ПРОГРАМУВАННЯ PYTHON

Донецький національний університет імені Василя Стуса, м. Вінниця

Вступ. Сортування злиттям у Python є одним з ефективних алгоритмів сортування, який базується на принципі «розділяй і володарюй». Цей алгоритм розбиває вхідний масив на менші частини, сортує їх окремо, а потім об'єднує в єдиний відсортований масив. Основні етапи алгоритму включають розділення, сортування та злиття, що гарантує ефективність і стабільність сортування.

Актуальність. Алгоритм сортування злиттям знайшов широке застосування в сучасному програмуванні, оскільки він вирішує проблему ефективного сортування великих обсягів даних. У сучасному світі, де обробка великих масивів даних стала нормою, ефективні алгоритми сортування є критичними для оптимальної роботи програм та додатків. Алгоритм Merge Sort вирізняється своєю асимптотичною оптимальністю з часовою складністю O(n log n), що робить його хорошим вибором для великих обсяг даних, порівняно з іншими алгоритмами сортування. Подальша оптимізація алгоритму, зокрема використання сортування вставкою для невеликих частин масиву та уникання зайвих операцій копіювання під час злиття, може ще більше підвищити його продуктивність у реальних випадках використання. Розгляд алгоритму та його оптимізацій здебільшого важливий для розробників, що працюють із великими обсягами даних і прагнуть досягти оптимальної продуктивності своїх програм.

Основні кроки алгоритму Merge Sort:

Розділення (Divide):

- вхідний масив розділяється на дві половини, знаходячи середину масиву (middle);
- рекурсивно застосовується алгоритм Merge Sort до кожної половини. Цей крок повторюється, поки розмір кожної половини не стає 1 або менше (базовий випадок).

***Сортування (Conquer)*:**

- кожна половина масиву сортується рекурсивно за допомогою алгоритму Merge Sort.

Злиття (Merge):

- два відсортованих підмасиви (ліва та права половини) об'єднуються в один відсортований масив;

➢ створюється новий порожній масив (result), який буде містити об’єднані елементи;

➢ ітеративно порівнюються елементи лівого та правого підмасивів;

➢ елемент, який має менше значення, додається до результату, а потім відповідний підмасив оновлюється, видаляючи доданий елемент;

➢ якщо один із підмасивів стає порожнім, залишок іншого підмасиву додається до результату;

➢ процес триває, поки обидва підмасиви стають порожніми;

➢ отриманий відсортований масив повертається.

Повторне застосування цих трьох кроків приведе до повного відсортування вхідного масиву.

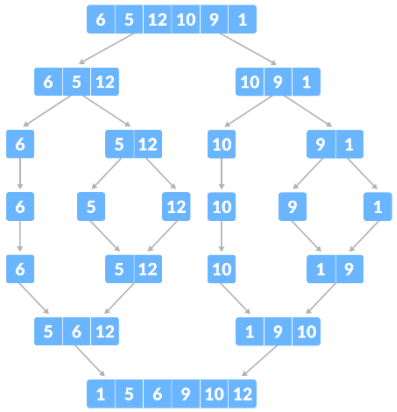

Рисунок 1. Приклад сортування злиттям

Швидкість алгоритму є асимптотично оптимальною, але її можна пришвидшити у константну кількість разів. Оптимізація впорядкування невеликих частин масиву – невеликі частини масиву (наприклад, кількістю менше 50) впорядковувати сортуванням вставкою. Оптимізація кількості копіювань елементів – під злиття двох упорядкованих масивів в один кожен елемент копіюється двічі (спочатку у тимчасовий масив, а потім знову у початковий). Кількість копіювань можна зменшити удвічі, якщо по черзі використовувати для об’єднання початковий і тимчасовий масиви.

Переваги алгоритму сортування злиттям:

➢ стабільність (алгоритм сортування злиттям є стабільним, що означає, що порядок рівнозначних елементів не змінюється. Це важливо в тих випадках, коли необхідно зберігати порядок елементів, які мають однакові ключі);

➢ асимптотична ефективність (у найгіршому, середньому та кращому випадках часова складність сортування злиттям становить $O(n \log n)$, де n – кількість елементів у масиві. Це робить його ефективним для великих об’ємів даних);

➢ послідовна реалізація (сортування злиттям може легко адаптуватися для паралельної обробки, що робить його практичним для використання в сучасних багатоядерних системах);

➢ невибагливість до пам'яті (хоча алгоритм вимагає додаткового простору для збереження тимчасових підмасивів під час злиття, його можна реалізувати так, щоб цей додатковий простір був константним (наприклад, за допомогою ітеративної версії замість рекурсивної).

Недоліки алгоритму сортування злиттям:

➢ неоптимальний для невеликих масивів (для невеликих масивів або вже частково відсортованих даних інші алгоритми, як-то сортування вставкою чи швидке сортування, можуть працювати швидше, оскільки витрачається менше часу на розділення та злиття);

➢ не змінює порядок рівнозначних елементів (якщо важливо змінювати порядок рівнозначних елементів для певних вимог, то сортування злиттям може не бути найкращим вибором);

➢ споживання часу на рекурсію (рекурсивному варіанті алгоритму може бути деякий витрати часу на виклик функцій та збереження додаткового контексту, особливо за великих об'ємах даних.)

Висновки. Сортування злиттям у Python є потужним і ефективним алгоритмом, заснованим на принципі «розділяй і володарюй». У статті ми розглянули основні етапи алгоритму: розділення, сортування та злиття, і висвітлили його ключові переваги.

Актуальність сортування злиттям визначається його ефективністю у роботі з великими обсягами даних, що є актуальним у сучасному програмуванні. Захист порядку рівнозначних елементів та асимптотична оптимальність часової складності роблять його привабливим вибором для розробників, які працюють із великими об'ємами даних та прагнуть досягти оптимальної продуктивності своїх програм.

Оптимізація алгоритму, а саме використання сортування вставкою для невеликих частин масиву та уникання зайвих операцій копіювання під час злиття, посилюють його продуктивність.

У галузі сортування й оптимізації алгоритмів сортування злиттям – є потужним інструментом, який відповідає високим вимогам ефективності та стабільності під час обробки великих обсягів даних.

Список використаних джерел

1. Introduction to Algorithms, 3-тє видання / Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. MIT Press. 2009.

2. Merge Sort. URL: https://www.geeksforgeeks.org/merge-sort/ (дата звернення: 13.11.2023).