Марчук Г. В., ст. викладач,
Ковальчук О. М., здобувач,
Державний університет «Житомирська політехніка»

ЧАСОВА СКЛАДНІСТЬ АЛГОРИТМІВ СОРТУВАННЯ

Алгоритм сортування – це процес впорядкування множини однотипних елементів за певною ознакою. Сортування важливе для оптимізації ефективності інших алгоритмів, які вимагають, щоб вхідні дані були у відсортованих списках. Одною з характеристик алгоритмів сортування є алгоритмічна складність.

Алгоритмічна складність – це кількісна характеристика, показник того, скільки часу знадобиться для завершення алгоритму з урахуванням вхідних даних розміром n. Основними характеристиками, що визначають складність алгоритмів є час, необхідний на впорядкування множини з n елементів і обсяг оперативної пам'яті необхідний для виконання сортування.

Часова складність алгоритму сортування описує кількість кроків, необхідних для його виконання, залежно від розміру вхідних даних.

Існує кілька способів оцінити часову складність:

- Асимптотична складність. Оцінює поведінку алгоритму при збільшенні розміру даних до нескінченності.
- Середня складність – це середня кількість кроків, необхідних для сортування даних довільного розміру.
- Найгірша складність – це максимальна кількість кроків, які алгоритм сортування може виконати для сортування масиву довільного розміру.

Найпоширеніші позначення часової складності:

- Складність постійного часу *O(1).*На алгоритм не впливає розмір вхідних даних. Для завершення певної операції потрібна фіксована кількість кроків, і ця кількість не залежить від кількості вхідних даних.
- Лінійна складність O(n). Час виконання алгоритму зростає лінійно, в залежності від кількості вхідних даних. Тобто час виконання алгоритму пропорційний розміру даних.
- Логарифмічна складність O(log n). Щоб виконати певну операцію над n елементами, потрібно log(n) кроків, де основа логарифму зазвичай дорівнює 2.
- *O(n log n)* складність часу, де n – велике число.
- Квадратична часова складність *O(n^2).*Алгоритм виконується за логарифмічний час, якщо час його виконання пропорційний квадрату розміру вхідних даних.
- *O(n!)* – час виконання пропорційний факторіалу розміру даних.

Наведемо приклади часової складності популярних алгоритмів сортування:

- Сортування вибором, бульбашкове або вставкою – $O(n^2)$.
- Сортування злиттям, швидке або пірамідальне – *O(n log n)*

Проведемо дослідження часової складності наступних алгоритмів сортування:

- Бульбашкове сортування – простий алгоритм, що базується на порівнянні сусідніх елементів і їх обміні в разі потреби. Продовжується цей процес до тих пір, поки весь масив не буде відсортований. Бульбашкове сортування є одним з найлегших алгоритмів, але він не такий ефективний для великої кількості елементів.
- Сортування вибором – алгоритм сортування, який знаходить мінімальний (максимальний) елемент у списку та переміщує його на початок. Потім алгоритм повторює цей процес для решти списку, зменшуючи розмір списку на один елемент на кожній ітерації.
- Сортування вставками – алгоритм сортування, який вставляє кожний елемент масиву у його правильне місце.
- Сортування Шелла – алгоритм сортування, який є модифікацією сортування вставками. Він має на меті покращити його ефективність, особливо для великих масивів даних.

Реалізуємо сортування обміном, вибором, вставкам та Шелла для масиву дійсних чисел за спаданням, та порівняємо їх за швидкодією (Табл. 1).

Таблиця 1

Порівняння алгоритмів сортування за часом

К-сть елементів	BubbleSort, час	SelectionSort, час	InsertionSort, час	Shellmethod, час
25	0.001000	0.003000	0.002000	0.002000
1000	0.030000	0.026000	0.027000	0.026000

З таблиці 1 бачимо, що алгоритм BubbleSort, досить швидко сортує масив з 25 елементів, але час збільшується, якщо збільшується розмір масиву до 1000 елементів. Аналогічним чином збільшується час сортування у SelectionSort, InsertionSort і Shellmethod хоча час сортування на масиві з 1000 елементів у наведених алгоритмах краще ніж у BubbleSort. Для всіх алгоритмів часова складність буде дорівнювати: O(n^2) = O(25^2) = 625, O(n^2) = O(1000^2) = 1 000 000.

У алгоритмі Шелла використовується фіксований крок, який зазвичай дорівнює половині попереднього кроку. Наприклад, якщо початковий крок дорівнює N/2 (де N – розмір масиву), наступні кроки будуть N/4, N/8, ..., 1.

Тоді часова складність цього методу може бути оцінена як O(N^3/2). O(n^2) = O(1000^3/2) = 31622,78.

Важливо зазначити, що часова складність алгоритму може варіюватися залежно від:

- *Типу даних.* Тип даних (цілі числа, дійсні числа, рядки) значно впливає на алгоритм сортування. Наприклад, для сортування великих масивів цілих чисел краще використовувати швидке сортування, а для великих масивів дійсних чисел краще використовувати сортування злиттям.
- *Розміру масиву даних.* Деякі алгоритми, такі як сортування вибором, можуть бути більш ефективними для невеликих масивів даних, швидке сортування - для великих, сортування злиттям, можуть бути більш ефективними для вже частково відсортованих даних.
- *Реалізації алгоритму.* Мова програмування, використовувані бібліотеки, різні реалізації одного і того ж алгоритму можуть мати різну часову складність.
- *Апаратне забезпечення.* Процесор, пам'ять.

Важливо знати часову складність алгоритму сортування, щоб зробити правильний вибір алгоритму для конкретної задачі.

При виборі алгоритму сортування важливо враховувати:

- *Розмір вхідних даних*. Для невеликих наборів даних не важливо, який алгоритм використовувати. Найкращим може бути алгоритм з більш простою реалізацією.
- *Тип даних*. Деякі алгоритми сортування більш ефективні для певних типів даних.
- *Апаратне забезпечення*. Деякі алгоритми сортування більш ефективні на певних типах апаратного забезпечення.
- *Складність даних*. Деякі алгоритми сортування більш ефективні для даних, які вже частково впорядковані.

Список використаних джерел

Трофименко, Р.Д. Аналіз впливу мов програмування на швидкодію алгоритмів сортування. *21 th–23 th September, 2023, Odessa*, p.208.