Практична робота №3.

2025.10.12,м.Кременьчук	Створив: Огоновський О.Є.

Мета: Опанувати основні алгоритми сортування та навчитись методам аналізу їх асимптотичної складності.

Завдання 1

Вивчити самостійно і записати (будь-яким способом) алгоритм бульбашкового сортування. Оцінити асимптотику алгоритму сортування методом бульбашки в найгіршому і в найкращому випадку. Порівняти за цими показниками бульбашковий алгоритм з алгоритмом сортування вставлянням. Чому на практиці бульбашковий алгоритм виявляється менш ефективним у порівнянні з сортуванням методом зливанням?

Алгоритм

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

Асимптотична складність

Якщо список відсортовано навпаки або невідсортовано: Кількість операцій: n^2, Складність: $O(n^2)$

Якщо список відсортований: Кількість операцій: n, Складність: O(n)

Порівняння з сортуванням вставками

Характеристика	Bubble Sort	Insertion Sort
Складність найгірший випадок	$O(n^2)$	$O(n^2)$
Складність найкращій випадок	$O(n)$	$O(n)$

Чому на практиці бульбашковий алгоритм виявляється менш ефективним у порівнянні з сортуванням методом зливанням?

в найгіршому випадку у бульбашки це O(n^2), а у злиття це O(nlog n).

Завдання 2

Оцінити асимптотичну складність алгоритму сортування зливанням, скориставшись основною теоремою рекурсії.

Рекурсивне рівняння

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

Основна теорема рекурсії

a = 2, b = 2, f(n) = O(n) Оскільки $f(n) = \Theta(n) = \Theta(n^{log_b(a)})$, то $T(n) = \Theta(nlogn)$

Завдання 3

Вивчити і записати (будь-яким способом) самостійно алгоритм швидкого сортування. Оцінити асимптотичну складність алгоритму швидкого сортування, скориставшись основною теоремою рекурсії.

```python
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)
```

$$T(n) = 2T(\frac{n}{2}) + O(n)\, T(n) = aT(\frac{n}{b}) + O(n^d)\, T(n) = \Theta(nlogn)$$

Асимптотична складність у середньому і найкращому випадку: O(n log n)

У найгіршому випадку: O(n^2)

Контрольні питання

1. Що таке асимптотична складність алгоритму? Асимптотична складність — це спосіб оцінки ефективності алгоритму шляхом аналізу того,як час виконання (часова складність) або обсяг пам'яті (просторова складність) змінюються зі збільшенням розміру вхідних даних.

2. Які алгоритми сортування мають квадратичну складність у найгіршому випадку? Поясніть, чому це може бути проблемою для великих обсягів даних.

   Сортування бульбашкою , сортування вставкою.Квадратична складність алгоритму означає , що час виконання алгоритму зростає пропорційно квадрату кількості елементів у списку.Для великих обсягів даних це може призвести до неприйнятно тривалого часу виконання.

3. В чому полягає перевага сортування злиттям над сортуванням вставками для великих наборів даних?

   швидкість у великих обсягах даних , тому що сортування злиттям має асимптотичну складність O(nlog n), а сортування вставкою O(n^2).

4. Які алгоритми сортування використовуються для сортування списків у стандартних бібліотеках мов програмування, таких як Python, Java або C++?

   TimSort(Python),Introsort(C++),Dual-Pivot Quicksort(Java).

5. Яка різниця між алгоритмами сортування злиттям і швидким сортуванням? У яких випадках краще використовувати кожен з цих алгоритмів?

   Сортування злиттям гарантує O(n log n) складність, розбиваючи та зливаючи відсортовані частини, що ефективно для великої кількості даних. Швидке сортування, в середньому, також O(n log n), але може мати O(n^2) в гіршому випадку, хоча на практиці воно часто швидше, особливо для "на місці" сортування. Сортування злиттям стабільніше за швидке.

6. Які фактори слід враховувати при виборі алгоритму сортування для конкретної задачі?

   об'єм даних , точність , швидкість.