Звіт до практичної роботи №1

Тема: Алгоритми сортування та їх складність. Порівняння алгоритмів сортування.

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

Хід роботи

1.
```
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]
    return arr
```

Сортування вставленням
Найгірший випадок O(n^2)
Найкращий випадок O(n)

Сортування бульбашкою

Найгірший випадок O(n^2)

Найкращий випадок O(n)

Алгоритм бульбашкового сортування є менш ефективним, ніж сортування зливанням через свою квадратичну асимптотику в найгіршому випадку, в той час як сортування зливанням має лінійно-логарифмічну складність, що значно швидше для великих масивів.

2. T(n)=2T (2n)+O(n)
   Log(2) 2=1
   T(n)=O (nlog2 2logn)=O(nlogn)

3.
```
def quick_sort(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 quick_sort(left) + middle + quick_sort(right)
```

$T(n)=2T(n2)+O(n)$T(n)=2T(2n)+O(n)

Log b a=log(2) 2=1

$T(n)=O(n\log_{f_0} n)$T(n)=O(nlogn)

Контрольні запитання

1. Асимптотична складність алгоритму сортування - це оцінка часу або просторової складності алгоритму при зростанні розміру вхідних даних. Вона вказує, як швидко або ефективно алгоритм буде працювати при великих обсягах даних.
2. **Бульбашкове сортування**
   Сортування вибором
   Сортування вставками
   Квадратична складність у найгіршому випадку означає, що час виконання алгоритму зростає квадратично з розміром вхідних даних. Це може стати проблемою для великих обсягів даних, оскільки зі збільшенням розміру масиву час виконання цих алгоритмів значно збільшується. Наприклад, для списків з тисячами або мільйонами елементів квадратичні алгоритми сортування можуть працювати надто повільно, що неприйнятно для багатьох практичних застосувань.
3. Сортування злиттям має лінійно-логарифмічну складність $O(n\log_{f_0} n)$O(nlogn) у всіх випадках, що робить його ефективнішим для великих наборів даних порівняно з сортуванням вставками, яке має квадратичну складність $O(n2)$O(n2) у найгіршому випадку.
4. Pythom – Timsort, C++ - Introsort, Java - Dual-Pivot Quicksort
5. Основна різниця між сортуванням злиттям і швидким сортуванням полягає у підходах до розподілу та обробки елементів. Сортування злиттям краще використовувати для великих наборів даних, коли потрібно гарантувати стабільну асимптотичну складність та можливість сортування об'єктів з невеликим обсягом пам'яті. Швидке сортування зазвичай ефективніше для невеликих та середніх наборів даних, а також коли важливо мінімізувати використання додаткової пам'яті. Однак варто враховувати можливий найгірший випадок O*(n2)O(n*2) і обирати опорний елемент ретельно.
6. Розмір вхідних даних, швидкодія алгоритму, вимоги до використання пам'яті, стабільність сортування, вартість сортування в найгіршому випадку, особливості даних

Висновок: На цьому занятті я навчився основним алгоритмам сортування та методам аналізу їх асимптотичної складності.

Роботу підготував
Гладкий Іван