Міністерство освіти і науки України

Кременчуцький національний університет   
імені Михайла Остроградського

Навчально-науковий інститут електричної інженерії   
та інформаційних технологій

Кафедра автоматизації та інформаційних систем

НаВчальна дисципліна  
«**аЛГОРИТМИ І СТРУКТУРИ ДАННИХ**»

Звіт

З ПРАКТИЧНОЇ роботи №3

Виконав

студент групи КН-24-1

Озівський В. В.

Перевірив

доцент кафедри КІЕ

Сидоренко В. М.

Кременчук 2025

|  |  |
| --- | --- |
| Тема: | Алгоритми сортування та їх складність. Порівняння алгоритмів сортування |
| Мета: | опанувати основні алгоритми сортування та навчитись методам аналізу їх асимптотичної складності. |

Хід роботи

Постановка завдання

Виконати індивідуальне завдання. Завдання полягає у розв’язанні всіх наведених задач.

Розв’язання задач

### Алгоритм бульбашкового сортування

Алгоритм:

1. Послідовно порівнюємо сусідні елементи масиву.
2. Якщо перший елемент більший за другий, міняємо їх місцями.
3. Повторюємо процес для всього масиву, поступово «спливаючи» найбільший елемент у кінець.
4. Продовжуємо ітерації до повного впорядкування масиву.

Код на C++:

void bubbleSort(int arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                std::swap(arr[j], arr[j + 1]);

            }

        }

    }

}

Асимптотична складність:

Найгірший випадок: (масив відсортований у зворотному порядку).

Найкращий випадок: (масив уже відсортований, одна оптимізація може зменшити кількість ітерацій).

Порівняння з сортуванням вставками:

Обидва алгоритми мають у найгіршому випадку, але сортування вставками ефективніше на майже відсортованих даних ( у найкращому випадку).

Чому бульбашкове сортування менш ефективне за сортування злиттям?

Метод злиття працює за у всіх випадках, тоді як бульбашкове сортування - , що значно гірше для великих обсягів даних.

### Асимптотична складність сортування злиттям

Основна ідея алгоритму:

1. Рекурсивно розділити масив навпіл, поки не отримаємо підмасиви з 1 елемента.
2. Об’єднувати ці підмасиви у відсортованому порядку.

Рекурентне співвідношення:

Застосовуючи основну теорему рекурсії:

1. Алгоритм швидкого сортування

Алгоритм:

1. Вибираємо опорний елемент (pivot).
2. Розподіляємо елементи так, щоб всі менші за pivot були ліворуч, а більші - праворуч.
3. Рекурсивно застосовуємо швидке сортування до обох частин.

Код на C++:

int partition(int arr[], int low, int high) {

    int pivot = arr[high];

    int i = low - 1;

    for (int j = low; j < high; j++) {

        if (arr[j] < pivot) {

            i++;

            std::swap(arr[i], arr[j]);

        }

    }

    std::swap(arr[i + 1], arr[high]);

    return i + 1;

}

void quickSort(int arr[], int low, int high) {

    if (low < high) {

        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

Аналіз складності:

Найгірший випадок: (якщо вибирати поганий pivot, наприклад, найменший або найбільший елемент у вже відсортованому масиві).

Середній і найкращий випадок: , якщо баланс розбиття хороший.

Рекурентне рівняння:

За основною теоремою рекурсії отримуємо .

Переваги:

Використовує менше пам’яті, ніж сортування злиттям.

У середньому працює швидше, ніж сортування злиттям, завдяки кращій локальності даних у пам’яті.

Недоліки:

У найгіршому випадку працює повільніше, ніж сортування злиттям.

Не завжди стабільний (не зберігає порядок однакових елементів).

Відповіді на контрольні питання

1. Асимптотична складність - це оцінка швидкості зростання часу виконання алгоритму при збільшенні розміру вхідних даних. Вона важлива, бо допомагає вибрати найефективніший алгоритм для конкретних задач.
2. Квадратичну складність у найгіршому випадку мають бульбашкове сортування, сортування вибором, сортування вставками. Це проблема для великих обсягів даних, оскільки час виконання зростає дуже швидко.
3. Сортування злиттям має у всіх випадках, тоді як сортування вставками – у найгіршому випадку. Це робить злиття кращим вибором для великих наборів даних, особливо якщо вони не майже відсортовані.
4. У стандартних бібліотеках мов програмування:

Python - Timsort (гібрид швидкого сортування та злиття).

Java - Arrays.sort() використовує Dual-Pivot Quicksort або MergeSort.

C++ - std::sort() використовує Introsort (гібрид Quicksort, Heapsort і MergeSort).

1. Різниця між сортуванням злиттям і швидким сортуванням:

MergeSort - завжди , стабільний, але потребує додаткової пам’яті.

QuickSort - швидший у середньому випадку, але може мати без оптимізацій.

1. Фактори вибору алгоритму:
2. Розмір вхідних даних.
3. Чи потрібно зберігати порядок рівних елементів (стабільність).
4. Обмеження по пам’яті (MergeSort вимагає додаткової пам’яті).
5. Чи масив майже відсортований (InsertionSort підходить).
6. Часова складність у середньому та найгіршому випадку.