**Міністерство освіти і науки України**

**Львівський національний університет імені Івана Франка**

**Факультет прикладної математики та інформатики**

Кафедра програмування

Лабораторна робота №1

**АЛГОРИТМИ СОРТУВАННЯ**

з курсу “Алгоритми та структури даних”

Виконала:

студентка групи ПМІ-13 Демко Сніжана Іванівна

Львів – 2023

**Сортування вставками**

**Складність роботи алгоритму:**

* найкраща: 𝑂(𝑛)
* середня: 𝑂(𝑛2)
* найгірша: 𝑂(𝑛2)

**Просторова складність:** 𝑂(𝑛) **Стабільність алгоритму:** стабільний

**Алгоритм:**

1. Вважаємо, що перший елемент масиву вже відсортований.
2. Вибираємо наступний елемент з невідсортованої частини масиву. Назвемо цей елемент поточним.
3. Порівнюємо поточний елемент з елементом відсортованої частини масиву, починаючи з останнього.
4. Якщо елемент відсортованої частини більший за поточний, ми зсуваємо його праворуч на одну позицію , щоб звільнити місце для вставки поточного елемента.
5. Після того як знайдено місце для вставки поточного елемента, ми вставляємо його в цю позицію.
6. Повторюємо кроки 2-4 для кожного невідсортованого елемента у масиві.

**Приклад №1**

Дано: 𝑎𝑟𝑟 = {5, 3, -4, 10, 6, 0}, 𝑠𝑖𝑧𝑒 = 6 .

Зображення, що містить текст, квитанція, Шрифт, білий

Автоматично згенерований опис

**Приклад №2**

Дано: 𝑎𝑟𝑟 = {40, 5, -10, 7, 17,-2, 3}, 𝑠𝑖𝑧𝑒 = 7.

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

**Висновок:** алгоритм сортування вставками є ефективний для невеликих масивів або масивів з переважно відсортованими елементами. Це хороший варіант для вивчення та засвоєння базових концепцій сортування, оскільки його легко реалізувати, а логіка роботи проста. Однак алгоритм має квадратичну складність і стає неефективним для великих масивів, і в цьому випадку краще використовувати більш ефективні алгоритми сортування, такі як швидке сортування або злиття.

**Ниткоподібне сортування**

**Складність роботи алгоритму:**

* найкраща: 𝑂(𝑛)
* середня: 𝑂(𝑛2)
* найгірша: 𝑂(𝑛2)

**Просторова складність:** 𝑂(1)

**Стабільність алгоритму:** не стабільний

**Алгоритм:**

1. Створення підсписку та створення фінального списку.
2. Додаємо перший елемент з основного списку у підсписок.
3. Проходимось по іншим елементам основного списку зліва на право. Якщо наступний елемент більший або дорівнює останньому елементу в підсписку, то додаємо його в підсписок.
4. Коли ми перейшлися по усім елементам основного списку, то посортований підсписок з першої ітерації потрібно помістити в порожній фінальний список.
5. Повторюємо кроки 2-3
6. Оскільки фінальний список уже не порожній - злити підсписок та фінальний список.
7. Повторити кроки 4-5 допоки основний список не залишиться пустим.

**Приклад №1**

Дано: 𝑎𝑟𝑟 = { 52, 0, -7, 6, 1, 9, 4,- 3 }, 𝑠𝑖𝑧𝑒 = 8 .

Зображення, що містить текст, квитанція, Шрифт, знімок екрана

Автоматично згенерований опис

**Приклад №2**

Дано: 𝑎𝑟𝑟 = { 5, 3, 8, 0, 9, -2, 7, 4, 6 }, 𝑠𝑖𝑧𝑒 = 9 .

Зображення, що містить текст, Шрифт, знімок екрана, алгебра

Автоматично згенерований опис

**Висновок:** алгоритм ниткоподібного сортування доцільно використовувати з послідовностями, які містять велику кількість вже відсортованих підпослідовностей. Він відносно легкий у реалізації та динамічно адаптується до структури вхідних даних. Також варто врахувати, що цей алгоритм може бути нестабільним.