Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики

Кафедра дискретного аналізу та інтелектуальних систем

Лабораторна робота №1

АЛГОРИТМИ СОРТУВАННЯ

з курсу “Алгоритми та структури даних”

Виконав:
студент групи ПМІ-13
Лук’янчук Денис
Євгенійович

Львів – 2024

Ниткоподібне сортування

Складність роботи алгоритму:

- найкраща: $O(n)$
- середня: $O(n^2)$
- найгірша: $O(n^2)$

Просторова складність: $O(1)$

Стабільність алгоритму: нестабільний

Алгоритм:

1. Переглядаємо список і витягаємо відсортований підсписок, починаючи з першого елемента.

2. Поміщаємо відсортований підсписок з першої ітерації в порожній відсортований список.

3. Повторяємо крок 1.

4. Оскільки відсортований список більше не порожній, об’єднуємо підсписок і відсортований список.

5. Повторюємо кроки 3-4, доки не буде видалено всі елементи зі списку.

Приклад №1

Дано: $arr = \{5, 3, -4, 10, 6\}$

```
Initial Array: 5 3 -4 10 6

Current State: 5 10

Current State: 3 5 6 10

Current State: -4 3 5 6 10

Sorted Array: -4 3 5 6 10
```

Приклад №2

Дано: $arr = \{8, 7, 3, 0, 5, 12, 3, 2, 1\}$

```
Initial Array: 8 7 3 0 5 12 3 2 1

Current State: 8 12

Current State: 7 8 12

Current State: 3 5 7 8 12

Current State: 0 3 3 5 7 8 12

Current State: 0 2 3 3 5 7 8 12

Current State: 0 1 2 3 3 5 7 8 12

Sorted Array: 0 1 2 3 3 5 7 8 12
```

Висновок: Ниткоподібне сортування доцільно застосовувати для даних, що зберігаються у зв'язному списку, через часте додавання та вилучення елементів. Якщо використовувати іншу структуру даних — наприклад, масив, тоді час виконання та складність алгоритму значно зростають. Також це сортування варто використовувати, коли велика частина даних уже посортована, тому що такі дані вилучаються одною “ниткою”.

Сортування включенням

Складність роботи алгоритму:

- найкраща: $O(n)$
- середня: $O(n^2)$
- найгірша: $O(n^2)$

Просторова складність: $O(\mathrm{n})$

Стабільність алгоритму: стабільний

Алгоритм:

1. Починаємо з першого елемента як відсортованої частини, а решту – як невідсортованої частини. Для простоти вважайте, що перший елемент є відсортованим підмасивом розміру 1.
2. Починаємо ітерацію з другого елемента (індекс 1) до кінця масиву. Цей елемент буде тимчасово видалено з масиву та розміщено у правильному місці в межах відсортованого підмасиву.
3. Порівнюємо поточний елемент з елементами відсортованого підмасиву. Переміщуємо елементи у відсортованому підмасиві, які перевищують поточний елемент, праворуч. Припиняємо переміщення елементів, коли ми знайдемо елемент у відсортованому підмасиві, який менший або дорівнює поточному елементу. Вставляємо поточний елемент у правильну позицію у відсортованому підмасиві.
4. Повторюємо кроки 2 і 3, доки не досягнемо кінця невідсортованої частини масиву. На кожній ітерації відсортований підмасив збільшується, а невідсортована частина зменшується.

Приклад №1

Дано: $arr = \{4, 6, 0, 4, 10, 12, 8, 2, 3\}$

```
Initial Array: 4 6 0 4 10 12 8 2 3
Iteration 1: 4 6 0 4 10 12 8 2 3
Iteration 2: 0 4 6 4 10 12 8 2 3
Iteration 3: 0 4 4 6 10 12 8 2 3
Iteration 4: 0 4 4 6 10 12 8 2 3
Iteration 5: 0 4 4 6 10 12 8 2 3
Iteration 6: 0 4 4 6 8 10 12 2 3
Iteration 7: 0 2 4 4 6 8 10 12 3
Iteration 8: 0 2 3 4 4 6 8 10 12
Sorrted Array: 0 2 3 4 4 6 8 10 12
```

Приклад №2

Дано: $arr = \{ 20, -3, 7, 4, 13, -1, 6, -8, 7 \}$

```
Initial Array: 20 -3 7 4 13 -1 6 -8 7
Iteration 1: -3 20 7 4 13 -1 6 -8 7
Iteration 2: -3 7 20 4 13 -1 6 -8 7
Iteration 3: -3 4 7 20 13 -1 6 -8 7
Iteration 4: -3 4 7 13 20 -1 6 -8 7
Iteration 5: -3 -1 4 7 13 20 6 -8 7
Iteration 6: -3 -1 4 6 7 13 20 -8 7
Iteration 7: -8 -3 -1 4 6 7 13 20 7
Iteration 8: -8 -3 -1 4 6 7 7 13 20
Sorrted Array: -8 -3 -1 4 6 7 7 13 20
```

Висновок: Сортування включенням або сортування вставлянням - це простий алгоритм, оснований на порівняннях. Хоча він менш ефективний на великих масивах порівняно з швидким сортуванням, пірамідальним сортуванням та сортуванням злиттям, він має свої переваги. Зокрема, він простий у реалізації і ефективний на малих масивах. Також він демонструє хорошу продуктивність при сортуванні вже відсортованих даних. Навіть порівняно з іншими квадратичними алгоритмами, такими як сортування вибором та сортування бульбашкою, він практично ефективніший, особливо у найкращому випадку, де його швидкодія є лінійною. Крім того, сортування включенням є стабільним алгоритмом.