МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут комп’ютерних систем

Кафедра інформаційних систем

Звіт

Лабораторна робота №1

З дисципліни **«**Теорія алгоритмів**»**

**Тема:** **«Квадратичні алгоритми сортування»­**

Виконав:

Студент групи AI-191У

Ігнатюк А.Г

Перевірили:

Бабілунга О.Ю

Одеса ­­­­– 2020

**Тема:**  «Квадратичні алгоритми сортування»

**Мета роботи:** Розглянути різновиди квадратичних алгоритмів сортування: вибору та вставками. Порівняти використані алгоритми.

**План:**

1. Сортування алгоритмом вибору
2. Сортування алгоритмом вставок
3. Порівняння алгоритмів
4. Висновок
5. **Сортировка выбором**

Ниже представлен псевдокод данного алгоритма:

**(i = 0, j = 0)**

# main cycle

**for i in range(len(array)):**

**min\_index = i** # assigning a start value

# this cycle finds out the minimum value in the rest of the array

**for j in range(i+1, len(array)):**

# comparing elements to find the min index

**if array[j] < array[min\_index]: min\_index = j**

# placing found element to the right position

**temp = array[min\_index]**

**array[min\_index] = array[i]**

**array[i] = temp**

# done

Результат численного моделирования для массива :

array = [28, 76, 27, 10, 5, 35, 95, 16, 33]

1. i = 0, min\_index = 0, j = 1 : array[1] > array[0]
2. i = 0, min\_index = 0, j = 2 : array[2] < array[0], min\_index = 2
3. i = 0, min\_index = 2, j = 3 : array[3] < array[2], min\_index = 3
4. i = 0, min\_index = 3, j = 4 : array[4] < array[3], min\_index = 4
5. i = 0, min\_index = 4, j = 5 : array[5] > array[4]
6. i = 0, min\_index = 4, j = 6 : array[6] > array[4]
7. i = 0, min\_index = 4, j = 7 : array[7] > array[4]
8. i = 0, min\_index = 4, j = 8 : array[8] > array[4]
9. temp = array[4] = 5
10. array[4] = array[0] = 28
11. array[0] = temp = 5

[5, 76, 27, 10, 28, 35, 95, 16, 33]

1. i = 1, min\_index = 1, j = 2 : array[2] < array[1], min\_index = 2

…

18. i = 1, min\_index = 3, j = 8 : array[8] > array[3]

19. temp = array[3] = 10

20. array[3] = array[1] = 76

21. array[1] = temp = 10

[5, 10, 27, 76, 28, 35, 95, 16, 33]

22. i = 2, min\_index = 2, j = 3 : array[3] > array[2]

…

27. i = 2, min\_index = 7, j = 8 : array[8] > array[7]

28. temp = array[7] = 16

29. array[7] = array[2] = 27

30. array[2] = temp = 16

[5, 10, 16, 76, 28, 35, 95, 27, 33]

…

31. i = 3, min\_index = 3, j = 4 : array[4] < array[3], min\_index = 4

…

35. i = 3, min\_index = 7, j = 8 : array[8] > array[7]

36. temp = array[7] = 27

37. array[7] = array[3] = 76

38. array[3] = temp = 27

[5, 10, 16, 27, 28, 35, 95, 76, 33]

…

45. …

[5, 10, 16, 27, 28, 35, 95, 76, 33]

51. …

[5, 10, 16, 27, 28, 33, 95, 76, 35]

56. …

[5, 10, 16, 27, 28, 33, 35, 76, 95]

60. …

[5, 10, 16, 27, 28, 33, 35, 76, 95]

61. (i = 8) temp = array[8] = 95

62. array[8] = array[8] = 95

63. array[8] = temp = 95

На выполнение алгоритма потребовалось **63** действия

1. **Сортировка вставками**

Ниже представлен псевдокод данного алгоритма:

**(i = 0, j = 0)**

# main cycle

**for i in range(1, len(array)):**

**j = i-1** # defining a start position to compare with

**temp = array[i]** # the element we want to place in the right position

# this cycle finds a home for the element

**while j >= 0 and** array**[j] > temp:**

**array[j+1] = array[j]**

**j-=1**

**array[j+1] = temp** # putting remembered element to the position

# that was found in 'while' cycle

# done

Результат численного моделирования для массива :

array = [28, 76, 27, 10, 5, 35, 95, 16, 33]

1. i = 1, j = 0, temp = array[1] = 76, j >= 0 and array[0] < 76 skip
2. array[1] = temp = 76
3. i = 2, j = 1, temp = 27, j >= 0 and array[1] > 27 : array[2] = array[1], j-=1

[28, 76, 76, 10, 5, 35, 95, 16, 33]

1. i = 2, j = 0, temp = 27, j >= 0 and array[0] > 27 : array[1] = array[0], j-=1
2. array[0] = temp = 27

[27, 28, 76, 10, 5, 35, 95, 16, 33]

1. i = 3, j = 2, temp = 10, j >= 0 and array[2] > 10 : array[3] = array[2], j-=1
2. i = 3, j = 1, temp = 10, j >= 0 and array[1] > 10 : array[2] = array[1], j-=1
3. i = 3, j = 0, temp = 10, j >= 0 and array[0] > 10 : array[1] = array[0], j-=1
4. array[0] = temp = 10

[10, 27, 28, 76, 5, 35, 95, 16, 33]

1. i = 4, j = 3, temp = 5, j >= 0 and array[3] > 5 : array[4] = array[3], j-=1
2. i = 4, j = 2, temp = 5, j >= 0 and array[2] > 5 : array[3] = array[2], j-=1
3. i = 4, j = 1, temp = 5, j >= 0 and array[1] > 5 : array[2] = array[1], j-=1
4. i = 4, j = 0, temp = 5, j >= 0 and array[0] > 5 : array[1] = array[0], j-=1
5. array[0] = temp = 5

[5, 10, 27, 28, 76, 35, 95, 16, 33]

1. i = 5, j = 4, temp = 35, j >= 0 and array[4] > 35 : array[5] = array[4], j-=1
2. i = 5, j = 3, temp = 35, j >= 0 and array[3] < 35
3. array[4] = temp = 35

[5, 10, 27, 28, 35, 76, 95, 16, 33]

1. i = 6, j = 5, temp = 95, j >= 0 and array[5] < 95
2. array[6] = temp = 95
3. i = 7, j = 6, temp = 16, j >= 0 and array[6] > 16 : array[7] = array[6], j-=1
4. i = 7, j = 5, temp = 16, j >= 0 and array[5] > 16 : array[6] = array[5], j-=1
5. i = 7, j = 4, temp = 16, j >= 0 and array[4] > 16 : array[5] = array[4], j-=1
6. i = 7, j = 3, temp = 16, j >= 0 and array[3] > 16 : array[4] = array[3], j-=1
7. i = 7, j = 2, temp = 16, j >= 0 and array[2] > 16 : array[3] = array[2], j-=1
8. i = 7, j = 1, temp = 16, j >= 0 and array[1] < 16
9. array[2] = temp = 16

[5, 10, 16, 27, 28, 35, 76, 95, 33]

1. i = 8, j = 7, temp = 33, j >= 0 and array[7] > 33 : array[8] = array[7], j-=1
2. i = 8, j = 6, temp = 33, j >= 0 and array[6] > 33 : array[7] = array[6], j-=1
3. i = 8, j = 5, temp = 33, j >= 0 and array[5] > 33 : array[6] = array[5], j-=1
4. i = 8, j = 4, temp = 33, j >= 0 and array[4] < 33
5. array[5] = temp = 33

[5, 10, 16, 27, 28, 33, 35, 76, 95]

На выполнение алгоритма потребовалось **31** действие

1. **Сравнение алгоритмов**

Алгоритм сортировки выбором: На очередной итерации будем находить минимум в массиве после текущего элемента и менять его с ним, если надо.

Алгоритм сортировки вставками: На очередной итерации будем менять текущий элемент с предыдущим, пока они стоят в неправильном порядке.

Оба алгоритма относятся к квадратичным сортировкам и являются весьма простыми.

**4) Вивод:**

В работе было рассмотрено два вида квадратичных сортировок: сортировка выбором и вставками. Было выполнено численное моделирование приведенного псевдокода на основе массива, данного по варианту задания. В этом случае сортировка вставками оказалась намного быстрее сортировки выбором и выполнила поставленную задачу за меньшее число итераций.