

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1
по курсу «Алгоритмы и структуры данных»
Тема: Сортировка вставкой, выбором, пузырьковая
Вариант 7

Выполнила
Пожидаева Е.Р.

Проверил:

Санкт-Петербург
2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту.....	3
Задание №1. Сортировка вставкой [N баллов]	3
Задание №3. Сортировка вставкой по убыванию [N баллов].....	7
Задание №4. Линейный поиск [N баллов].....	10
Вывод.....	14

Задачи по варианту

Задание №1. Сортировка вставкой [N баллов]

Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива $A = \{31, 41, 59, 26, 41, 58\}$.

```
f = open('input.txt')
n = int(f.readline())
s = f.readline()
f.close()
x = ([int(i) for i in s.split()])

def insertion_sort(arr):
    for j in range(1, len(arr)):
        key = arr[j]
        i = j - 1
        while i >= 0 and arr[i] > key:
            arr[i + 1] = arr[i]
            i = i - 1
        arr[i + 1] = key
    return arr

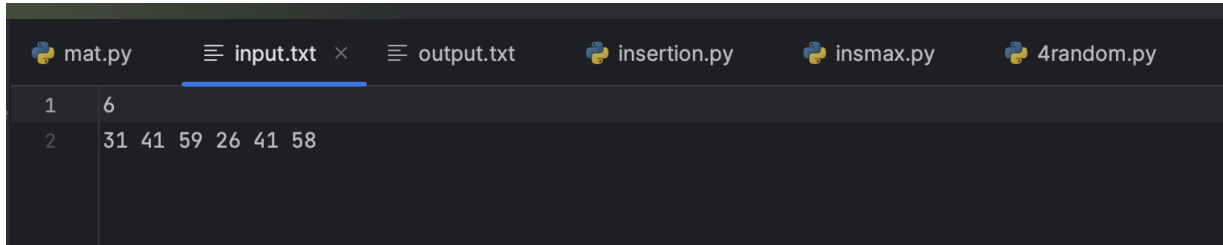
result = insertion_sort(x)
f = open('output.txt', 'w')
f.write(' '.join(map(str, result)))
f.close()
```

Текстовое объяснение решения.

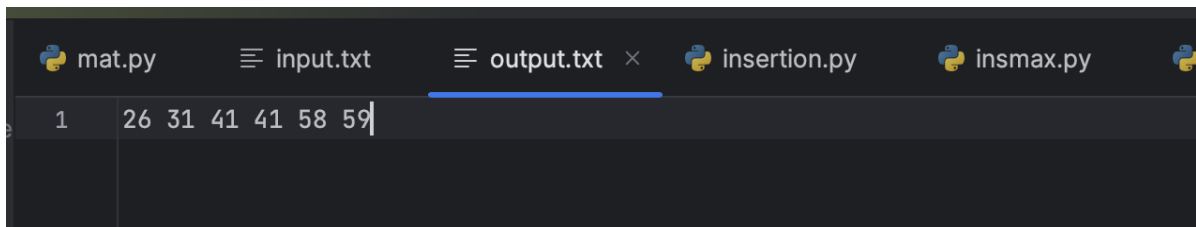
Первая строка файла input.txt записывается в переменную n, в ней хранится значение равное количеству чисел элементов в массиве. Вторая строка записывается в переменную s, в ней записаны сами элементы массива. Создаем массив x, в котором элементы из s, элементы делаем целочисленными. В функции insertion_sort запускается цикл с переменной j, которая будет от 1 до длины массива. Создаем переменную key = arr[j] Далее устанавливается, что $i = j - 1$. Запускается цикл while, который будет выполняться до тех пор, пока i не отрицательное и больше key. В этом цикле на место переменной, идущей после рассматриваемой переменной с индексом i ставится переменная с индексом i+1, а само i становится меньше на 1. После окончания цикла на место переменной с индексом i+1

записывается значение key. В итоге, функция возвращает отсортированный по возрастанию массив.

Результат работы кода на примере из текста задачи:



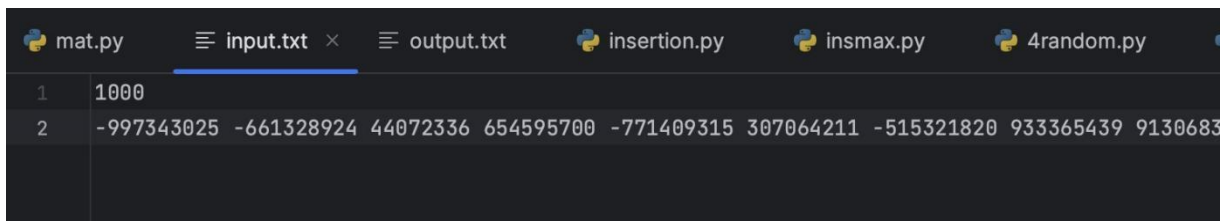
```
mat.py input.txt x output.txt insertion.py insmax.py 4random.py
1 6
2 31 41 59 26 41 58
```



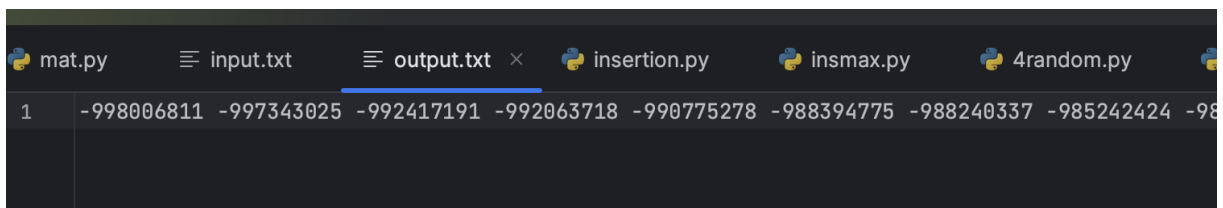
```
mat.py input.txt x output.txt insertion.py insmax.py
1 26 31 41 41 58 59
```

Результат работы кода на максимальных и минимальных значениях:

Максимальное:

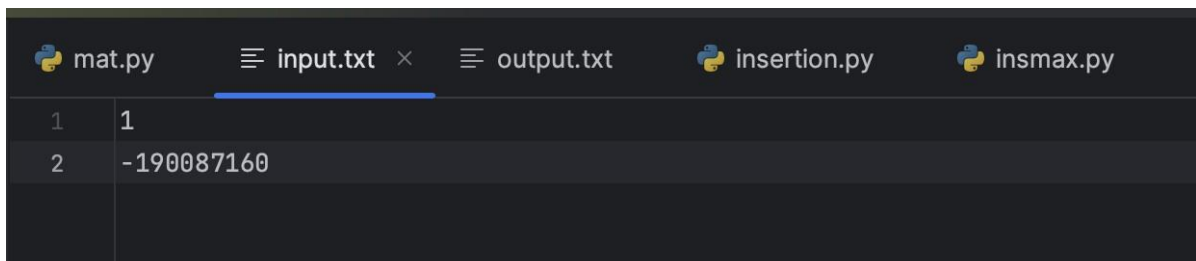


```
mat.py input.txt x output.txt insertion.py insmax.py 4random.py
1 1000
2 -997343025 -661328924 44072336 654595700 -771409315 307064211 -515321820 933365439 9130683
```

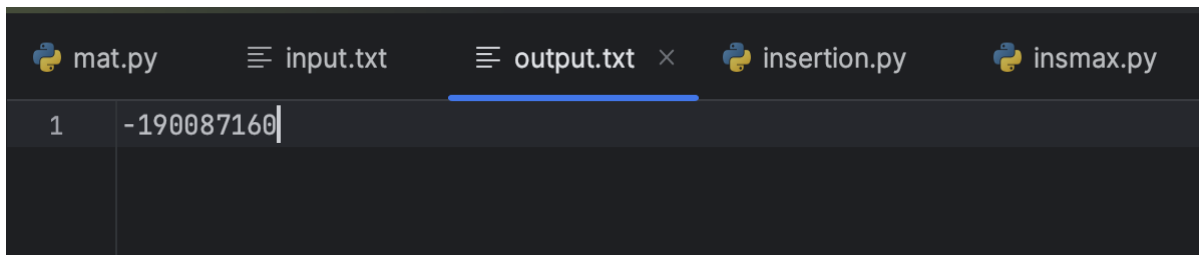


```
mat.py input.txt x output.txt insertion.py insmax.py 4random.py
1 -998006811 -997343025 -992417191 -992063718 -990775278 -988394775 -988240337 -985242424 -98
```

Минимальное:



```
mat.py input.txt x output.txt insertion.py insmax.py
1 1
2 -190087160
```



```
mat.py  input.txt  output.txt  insertion.py  insmax.py
1  -190087160
```

	Время выполнения, с	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.012844085693359375	6328320
Пример из задачи	0.0003418922424316406	6127616
Верхняя граница диапазона значений входных данных из текста задачи(1000 элементов)	0.06262397766113281	6238208

Выводы по задаче: Insertion-sort позволяет быстро отсортировать массив.

Задание №3. Сортировка вставкой по убыванию [N баллов]

Перепишите процедуру Insertion-sort для сортировки в невозрастающем порядке вместо неубывающего с использованием процедуры Swap.

```
f = open('input.txt')
n = int(f.readline())
s = f.readline()
f.close()
x = ([int(i) for i in s.split()])
```

```

def insertion_sort(arr):
    for j in range(1, len(arr)):
        key = arr[j]
        i = j - 1
        while i >= 0 and key > arr[i]:
            t = arr[i]
            arr[i] = key
            arr[i + 1] = t
            i = i - 1
    return arr

result = insertion_sort(x)
f = open('output.txt', 'w')
f.write(' '.join(map(str, result)))

```

Текстовое объяснение решения.

Первая строка файла input.txt записывается в переменную n, в ней хранится значение равное количеству чисел элементов в массиве. Вторая строка записывается в переменную s, в ней записаны сами элементы массива. Создаем массив x, в котором элементы из s, элементы делаем целочисленными. В функции insertion_sort запускается цикл с переменной j, которая будет от 1 до длины массива. Создаем переменную key = arr[j]. Далее устанавливается, что i = j - 1. Запускается цикл while, который будет выполняться до тех пор, пока i не отрицательное и элемент массива, индекс которого равен i, меньше переменной key. В этом цикле применяется процедура swap. Сначала в переменную t записывается значение arr[i], чтобы оно не было потеряно, затем на место arr[i] записывается объект key, а после на место arr[i+1] записывается значение t. Каждый цикл i уменьшается на 1, чтобы рассматривать все варианты. Итогом функции является возвращение отсортированного по убыванию массива A.

Результат работы кода на примере из текста задачи:

```
mat.py  input.txt  output.txt  insertion.py  insmax.p
1      6
2      31 41 59 26 41 58
```

```
mat.py  input.txt  output.txt  insertion.py  ins
1      59 58 41 41 31 26
```

Результат работы кода на максимальных и минимальных значениях:

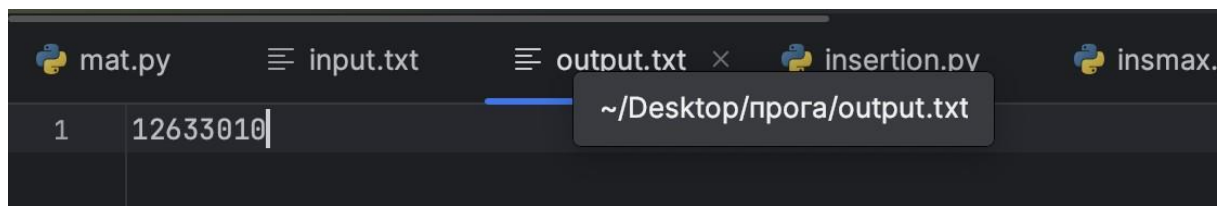
Максимальное:

```
mat.py  input.txt  output.txt  insertion.py  insmax.p
1      1000
2      -787159994 260991551 -314200550 505367748 945927710 647069424 1287
```

```
mat.py  input.txt  output.txt  insertion.py  insmax.p
1      998877111 998593782 994331014 991687362 989963769 989497718 989081
```

Минимальное:

```
mat.py  input.txt  output.txt  insertion.py  insm
1      1
2      12633010
```



```
mat.py input.txt output.txt x insertion.py insmax.  
1 12633010  
~/Desktop/порог/output.txt
```

	Время выполнения, с	Затраты памяти, Мб
Нижняя граница диапазона значений входных данных из текста задачи(1 элемент)	0.01673293113708496	6189056
Пример из задачи	0.00032901763916015625	6320128
Верхняя граница диапазона значений входных данных из текста задачи(1000 элементов)	0.07631397247314453	6242304

Выводы по задаче: С помощью процедуры swar можно выполнять сортировку в обратном порядке, по убыванию.

Задание №4. Линейный поиск [N баллов]

Рассмотрим задачу поиска.

- **Формат входного файла.** Последовательность из n чисел $A = a_1, a_2, \dots, a_n$ в первой строке, числа разделены пробелом, и значение V во второй строке. Ограничения: $0 \leq n \leq 10^3$, $-10^3 \leq a_i, V \leq 10^3$
- **Формат выходного файла.** Одно число - индекс i , такой, что $V = A[i]$, или значение -1 , если V в отсутствует.
- Напишите код линейного поиска, при работе которого выполняется сканирование последовательности в поисках значения V .
- Если число встречается несколько раз, то выведите, сколько раз встречается число и все индексы i через запятую.
- Дополнительно: попробуйте найти свинью, как в лекции. Используйте во входном файле последовательность слов из лекции, и найдите соответствующий индекс.

Листинг кода.

```
f = open('input.txt')
a = f.readline().split()
V = f.readline()

def search(n):
    b = []
    x = len(n)
    for i in range(x):
        if n[i] == V:
            b.append(i)
    if len(b) > 0:
        return len(b), b
    elif len(b) == 0:
        return '-1'

f1 = open('output.txt', 'w')
f1.write(' '.join(map(str, search(a))))
f1.close()
```

Текстовое объяснение решения.

Файл открывается для чтения, первая строка записывается в переменную `a`, вторая строка с элементом, который нам нужно найти, в переменную `V`. В функции создаем пустой массив `b`. С помощью цикла `for` проверяем все элементы списка. Если какой-то элемент равен тому, что мы ищем(`V`), то мы добавляем этот элемент в `b`. Если в итоге массив не пустой, то есть его длина больше 0, то функция возвращает длину массива (сколько раз элемент, который мы искали, встречается в файле) и сам массив. Если массив `b` пустой, то возвращается -1.

Результат работы кода на примере из текста задачи:

```
mat.py  input.txt  output.txt  insertion.py  insmax.py  insmin.py  ir
1  Рак Кабан Козел Лемминг Черепаха Муравьед Нутрия Индюк Ягуар Хамелеон Коала Барсук Акула ✓ 90
2  Нутрия
    
```

```
mat.py  input.txt  output.txt  insertion.py  insmax.py  insmin.py
1  1  [6]
    
```

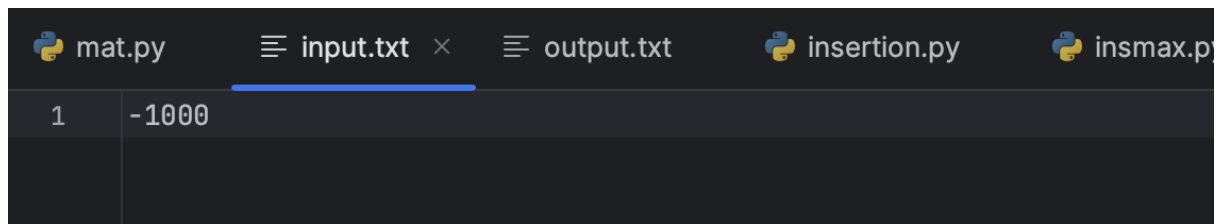
Результат работы кода на максимальных и минимальных значениях:

Максимальное:

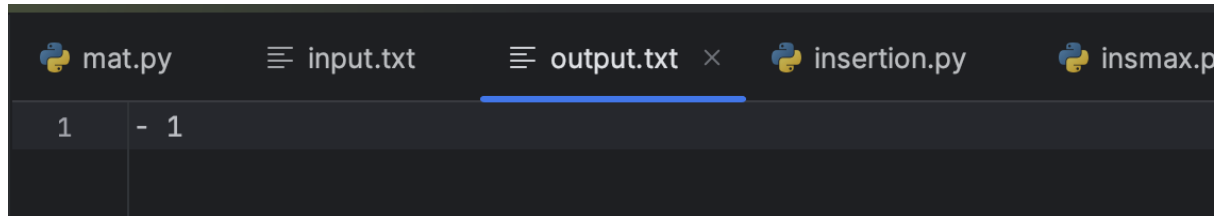
```
mat.py  input.txt  output.txt  insertion.py  insmax.py
1  -70 903 64 -316 -276 -466 988 -606 -550 -206 -201 -267 669 746 -60
2  114
    
```

```
mat.py  input.txt  output.txt  insertion.py  insmax.py
1  1  [214]
    
```

Минимальное:



The screenshot shows a code editor with four tabs: mat.py, input.txt, output.txt, insertion.py, and insmax.p. The input.txt tab is active and displays the number -1000.



The screenshot shows the same code editor with the output.txt tab active. It displays the result - 1.

	Время выполнения, с	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.0015120506286621094	6164480
Пример из задачи	0.0006091594696044922	6262784
Верхняя граница диапазона значений входных данных из текста задачи	0.014724969863891602	6254592

Вывод по задаче: Линейный поиск позволяет найти нужный элемент в большом списке.

Вывод по лабораторной работе №1:

В ходе данной лабораторной работы я узнала о различных методах сортировки.