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 вставками, выбором, пузырьковая
Вариант 22

Выполнил:
Чан Тхи Лиен
К3140

Проверил:
Афанасьев А.В.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задание 1. Сортировка вставкой

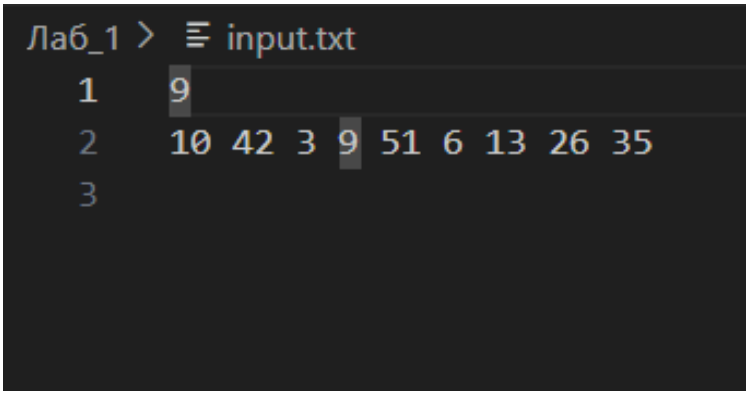
Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива $A = \{31, 41, 59, 26, 41, 58\}$.

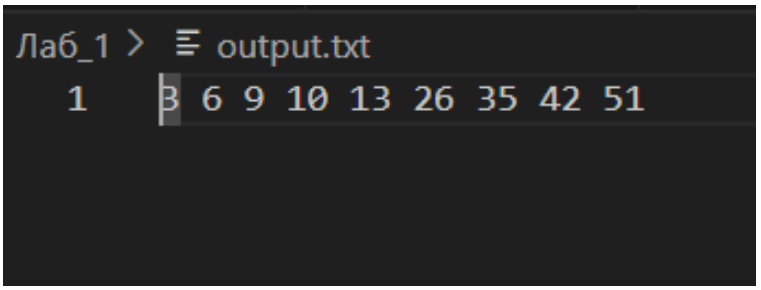
- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^3$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- **Формат выходного файла (output.txt).** Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

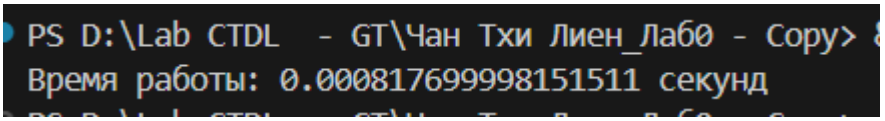
Выберите любой набор данных, подходящих по формату, и протестируйте алгоритм.

```
> task1 > task1.py > ...
# Insertion Sort in Python
#function
from time import perf_counter
def insertionSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]

        j = i-1
        while j >= 0 and key < arr[j] :
                arr[j + 1] = arr[j]
                j -= 1
        arr[j + 1] = key
#result
t1_start = perf_counter()
with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\input.txt','r') as infile:
    n = int(infile.readline())
    data = str(infile.readline())

data = data.split(' ')
arr = [int(i) for i in data]
if n!=len(data) or n<=0 or n>10**3:
    print('данные не верны')
else:
    insertionSort(arr)
    a = ' '.join(str(i) for i in arr)
    with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\output.txt', 'w') as outfile:
        outfile.write(a)
t1_stop = perf_counter()
print('Время работы: %s секунд '% (t1_stop - t1_start))
```

1. Импортирую библиотеку **time**
2. Использую **def** для создания функции
    - Цикл **for** чтобы создать программы, повторяющие действие
    - индекс начинается с 1 до n-1
    - Сохраняю элемент, который имеет позицию i в массиве, в **key**
    - Создаю j для поиска позиции вставки
    - Проверяю больше ли предыдущий элемент, чем текущий и сдвигаю элемент влево
    - Вставляю **key** в найденную позицию
    - Уменьшаю индекс j.
3. Использую **with open** чтобы открыть файл и **readline** для чтения каждой строки в файле
4. Метод **split() -** разделить строку на массив
5. Метод **join() -** собирает строку из элементов списка
6. Использую **with open** чтобы открыть файл и **write** для запись результаты в файле







Задание 6. Пузырьковая сортировка

Пузырьковая сортировка представляет собой популярный, но не очень эффективный алгоритм сортировки. В его основе лежит многократная перестановка соседних элементов, нарушающих порядок сортировки. Вот псевдокод этой сортировки:

```
Bubble_Sort(A):
  for i = 1 to A.length - 1
    for j = A.length downto i+1
      if A[j] < A[j-1]
        поменять A[j] и A[j-1] местами
```

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректоности процедуры вам необходимо доказать, что она завершается и что $A'[1] \leq A'[2] \leq \ldots \leq A'[n]$, где A' - выход процедуры Bubble_Sort, а n - длина массива A.

Определите время пузырьковой сортировки в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

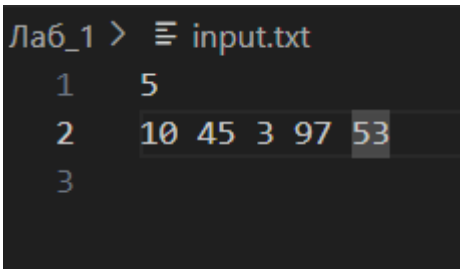
Формат входного и выходного файла и ограничения - как в задаче 1.

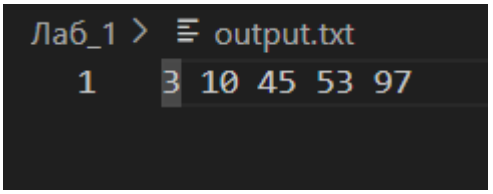
Task6 > Task6.py > ...

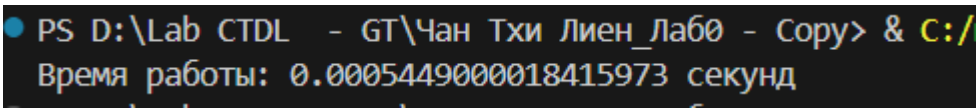
```python
# Bubble sort in Python
#function
from time import perf_counter
def bubbleSort(array):

  for i in range(len(array)):

    for j in range(0, len(array) - i - 1):

      if array[j] > array[j + 1]:

        temp = array[j]
        array[j] = array[j+1]
        array[j+1] = temp
#result
t1_start = perf_counter()
with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\input.txt','r') as infile:
    n = int(infile.readline())
    data = str(infile.readline())

data = data.split(' ')
arr = [int(i) for i in data]
if n!=len(data) or n<=0 or n>10**3:
    print('данные не верны')
else:
    bubbleSort(arr)
    a = ' '.join(str(i) for i in arr)
    with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\output.txt', 'w') as outfile:
        outfile.write(a)
t1_stop = perf_counter()
print('Время работы: %s секунд '% (t1_stop - t1_start))
```

Как Задание 1, но изменить функцию

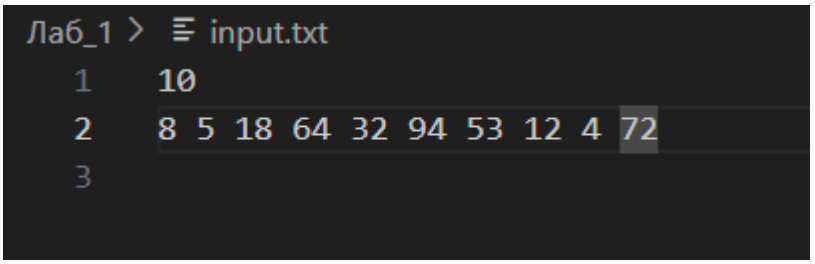
- Использую 2 повторяющегося действия
- Прохожу по всем элементам массива
- Если текущий элемент больше следующего, меняем их местами
- Используя эту функцию, мы расположим положение самого большого элемента к самому маленькому элементу.

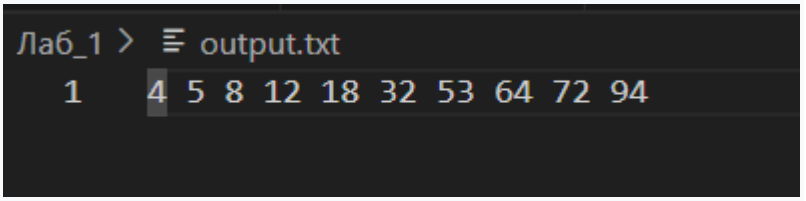


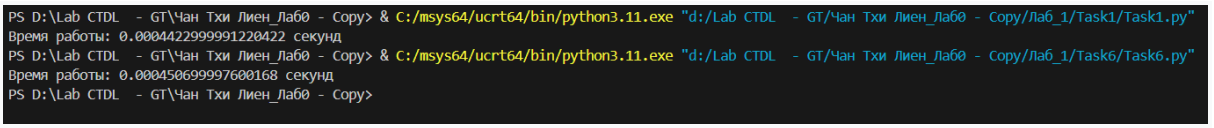




Использование функции **Bubble Sort** обычно занимает больше времени, чем **Insertion Sort**







Задание 7. Знакомство с жителями Сортлэнда

Владелец графства Сортлэнд, граф Бабблсортер, решил познакомиться со своими подданными. Число жителей в графстве нечетно и составляет n, где n может быть достаточно велико, поэтому граф решил ограничиться знакомством с тремя представителями народонаселения: с самым бедным жителем, с жителем, обладающим средним достатком, и с самым богатым жителем.

Согласно традициям Сортлэнда, считается, что житель обладает средним достатком, если при сортировке жителей по сумме денежных сбережений он оказывается ровно посередине. Известно, что каждый житель графства имеет уникальный идентификационный номер, значение которого расположено в границах от единицы до n. Информация о размере денежных накоплений жителей хранится в массиве M таким образом, что сумма денежных накоплений жителя, обладающего идентификационным номером i, содержится в ячейке $M[i]$. Помогите секретарю графа мистеру Свопу вычислить идентификационные номера жителей, которые будут приглашены на встречу с графом.

- **Формат входного файла (input.txt).** Первая строка входного файла содержит число жителей n ($3 \leq n \leq 9999$, n нечетно). Вторая строка содержит описание массива M, состоящее из положительных вещественных чисел, разделенных пробелами. Гарантируется, что все элементы массива M различны, а их значения имеют точность не более двух знаков после запятой и не превышают 10^6.

- **Формат выходного файла (output.txt).** В выходной файл выведите три целых положительных числа, разделенных пробелами — идентификационные номера беднейшего, среднего и самого богатого жителей Сортлэнда.

```
Task7 > Task7.py > ...
#function
def insertionSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]


        j = i-1
        while j >= 0 and key < arr[j] :
                arr[j + 1] = arr[j]
                j -= 1
        arr[j + 1] = key
#result
with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\input.txt','r') as infile:
    n = int(infile.readline())
    data = str(infile.readline())

data = data.split(' ')
arr = [float(i) for i in data]
list = [float(i) for i in data]
if n!=len(data) or n<=0 or n>10**3 or n%2 == 0:
    print('данные не верны')
else:
    insertionSort(arr)

    max = arr[-1]
    x = list.index(max)+1
    min = arr[0]
    y = list.index(min)+1
    tb = arr[n//2]
    z = list.index(tb)+1
    a = str(y)+' '+str(z)+' '+str(x)

    with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\output.txt', 'w') as outfile:
        outfile.write(a)
```

- Делаю как задание 1 чтобы расположить массив от меньшего к большему.
- **arr** - отсортированный массив

  list - исходный массив
- Беру самый большой, самый маленький и средний элемент в **arr**.
- затем используйте **index()** , чтобы найти их позицию в **list**

```
Лаб_1 > input.txt
1   5
2   10.00 8.70 0.01 5.00 3.00
3
```

```
Лаб_1 > output.txt
1   3 4 1
```

Дополнительные задачи

Задание 4. Линейный поиск

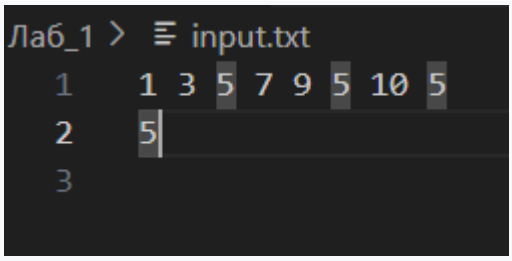
Рассмотрим задачу поиска.

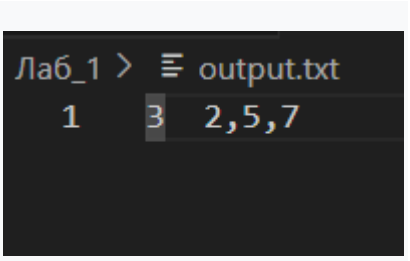
- **Формат входного файла**. Последовательность из n чисел $A = a_1, a_2, \ldots, a_n$ в первой строке, числа разделены пробелом, и значение V во второй строке. Ограничения: $0 \leq n \leq 10^3$, $-10^3 \leq a_i, V \leq 10^3$
- **Формат выходного файла**. Одно число - индекс i, такой, что $V = A[i]$, 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.

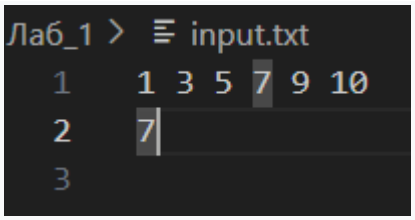
Task4 > Task4.py > ...

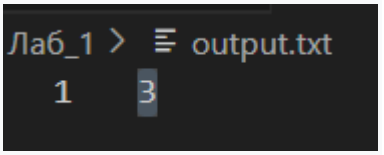
```python
#Linear Search in Python
def linearSearch(arr,key):
    list = []
    for i in range(len(arr)):
        if arr[i] == key:
            list.append(i)
    count = len(list)
    if count == 0:
        return str(-1)
    elif count == 1:
        return str(list[0])
    else:
        return str(count) + '  ' + ",".join(map(str, list))

with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\input.txt','r') as infile:
    data = str(infile.readline())
    key = int(infile.readline())

data = data.split(' ')
arr = [int(i) for i in data]
if len(data)<0 or len(data)>10**3 or -10**3>key or key>10**3:
    print('данные не верны')
else:

    with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\output.txt', 'w') as outfile:
        outfile.write(linearSearch(arr,key))
```

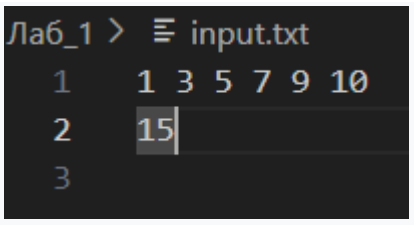
- Использую **list** для сохранения положения **key**
- **for -** Проверьте каждый элемент массива. Если элемент равен **key**, использую **append()** для запись положения **i** в **list**
- **len() -** подсчитать количество элементов в массиве **list.** Если количество равно 0, у массива нет **key** и записать -1 . Если равно 1, записать это положение. Если больше 1, записать количество и положения

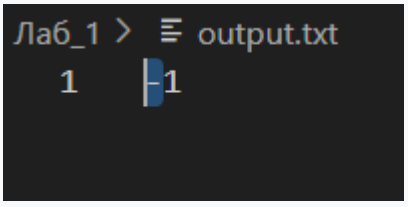












Задание 5. Сортировка выбором

Рассмотрим сортировку элементов массива , которая выполняется следующим образом. Сначала определяется наименьший элемент массива , который ставится на место элемента $A[1]$. Затем производится поиск второго наименьшего элемента массива A, который ставится на место элемента $A[2]$. Этот процесс продолжается для первых $n-1$ элементов массива A.

Напишите код этого алгоритма, также известного как сортировка выбором (selection sort). Определите время сортировки выбором в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

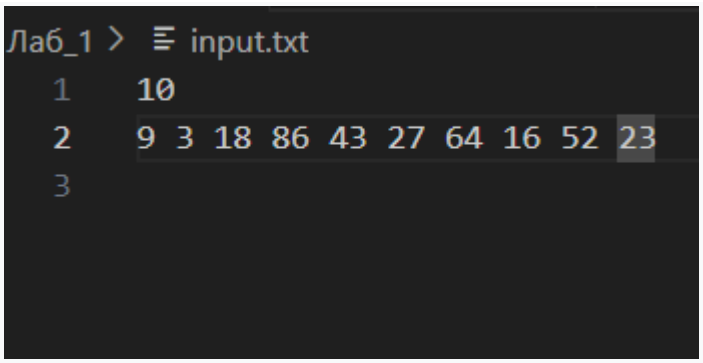
Формат входного и выходного файла и ограничения - как в задаче 1.

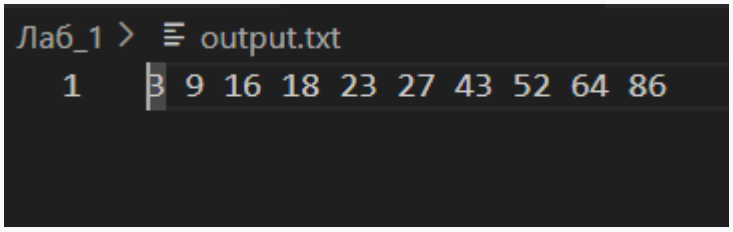
Лаб_1 > Task5 > Task5.py > ...

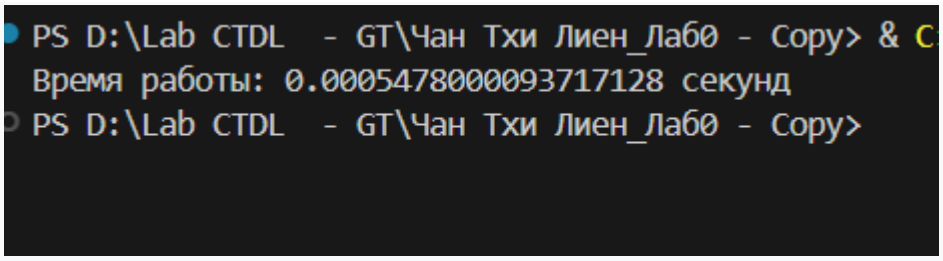
```python
# Selection Sort in Python
from time import perf_counter
#function
def selectionSort(array, size):

    for ind in range(size):
        min_index = ind

        for j in range(ind + 1, size):
            if array[j] < array[min_index]:
                min_index = j

        (array[ind], array[min_index]) = (array[min_index], array[ind])
#result
t1_start = perf_counter()
with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\input.txt','r') as infile:
    n = int(infile.readline())
    data = str(infile.readline())

data = data.split(' ')
arr = [int(i) for i in data]
size = len(arr)
if n!=len(data) or n<=0 or n>10**3:
    print('данные не верны')
else:
    selectionSort(arr, size)
    a = ' '.join(str(i) for i in arr)
    with open('D:\Lab CTDL  - GT\Чан Тхи Лиен_Лаб0 - Copy\Лаб_1\output.txt', 'w') as outfile:
        outfile.write(a)
t1_stop = perf_counter()
print('Время работы: %s секунд '% (t1_stop - t1_start))
```

Как Задание 1, но изменить функцию

- Использую 2 повторяющегося действия
- Прохожу по всем элементам массива
- Используя эту функцию, мы расположим положение самого маленького элемента к самому большому элементу.







Вывод

- Знакомство сортировки: **Insertion Sort, Bubble Sort, Selection Sort**
- Работа с файлом, у которого много строк и с массивом и строкой
- знать метод преобразования строки в массив и наоборот **split(), join()**
- добавлять элементы в массивы и строки **append()**
- Найти позицию элемента в массиве **index()**