

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.

Выполнил:
Серова С. А. (фамилия имя)
К3140 (номер группы)

Проверила:
—

Санкт-Петербург
2024 г.

Содержание отчета:

Вариант 20

Вариант	Номера задач	Вариант	Номера задач
1	1,2,4	16	1,4,8
2	1,2,5	17	1,4,9
3	1,2,6	18	1,5,6
4	1,2,7	19	1,5,7
5	1,2,8	20	1,5,8
6	1,2,9	21	1,5,9
7	1,3,4	22	1,6,7
8	1,3,5	23	1,6,8
9	1,3,6	24	1,6,9
10	1,3,7	25	1,7,8
11	1,3,8	26	1,7,9
12	1,3,9	27	1,8,9
13	1,4,5		
14	1,4,6		
15	1,4,7		

Задачи

Задача №1. Сортировка вставкой

Задача №3. Сортировка вставкой по убыванию

Задача №4. Линейный поиск

Задача №5. Сортировка выбором

Задача №8. Секретарь Своп

Задача №10. Палиндром

Задачи:

Задача №1. Сортировка вставкой

Текст задачи:

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.

Листинг кода:

```
import time
t_start = time.perf_counter()

def insertion_sort(second_line, first_line):
    for i in range(1, first_line):
        n = second_line[i]
        k = i - 1
        while (k >= 0 and n < second_line[k]):
            second_line[k + 1] = second_line[k]
            k = k - 1
        second_line[k + 1] = n

    return second_line

with open("input.txt", 'r') as file:
    first_line = int(file.readline().rstrip('\n'))
    second_line = (file.readline())
    second_line = second_line.split()
    second_line = [int(x) for x in second_line]

with open("output.txt", 'w') as file:
    a = insertion_sort(second_line, first_line)
    a = [str(s) for s in a]
    file.write(' '.join(a))

print("Время работы: %s секунд" % (time.perf_counter() - t_start))
```

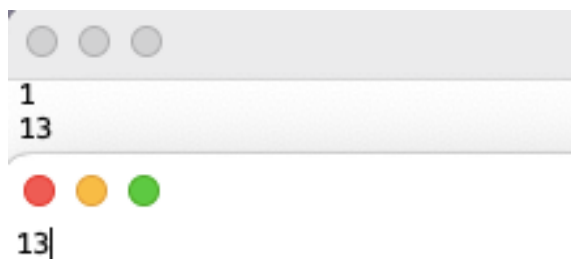
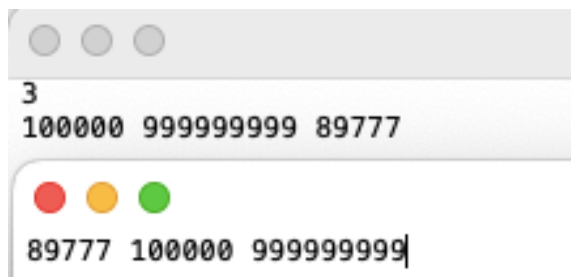
Текстовое объяснение решения:

Я считываю данные из файла, преобразую их в список. Внутри функции прохожу по списку и переставляю местами элементы (если элемент меньше предыдущих, то он сдвигается влево в списке). Также, преобразую элементы списка в строку и результат добавляю в файл.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов):



Время выполнения:

```
Время работы: 0.005074595013866201 секунд
```

Нижняя граница диапазона значений входных данных из текста задачи:

$1 \leq n$ (количество элементов)
 -10^{**9}

Верхняя граница диапазона значений входных данных из текста задачи:

n<=10**3 (количество элементов)
10**9

Вывод по задаче:

Я научилась сортировать вставкой данные из файла.

Задача №3. Сортировка вставкой по убыванию

Текст задачи:

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

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

Подумайте, можно ли переписать алгоритм сортировки вставкой с использованием рекурсии?

Листинг кода:

```
import time
t_start = time.perf_counter()

def swap(second_line, first_line):
    for i in range(1, first_line):
        n = second_line[i]
        j = i - 1
        while (j >= 0 and n > second_line[j]):
            second_line[j + 1] = second_line[j]
            j = j - 1
        second_line[j + 1] = n

    return second_line

with open("input.txt", 'r') as file:
    first_line = int(file.readline().rstrip('\n'))
    second_line = (file.readline())
    second_line = second_line.split()
    second_line = [int(x) for x in second_line]
with open("output.txt", 'w') as file:
    a = swap(second_line, first_line)
    a = [str(s) for s in a]
    file.write(' '.join(a))

print("Время работы: %s секунд" % (time.perf_counter()-
t_start))
```

Текстовое объяснение решения:

Я считываю данные из файла, преобразую их в список. Внутри функции прохожу по списку и переставляю местами элементы (если элемент больше предыдущих, то он сдвигается влево в списке). Также, преобразую элементы списка в строку и результат добавляю в файл.

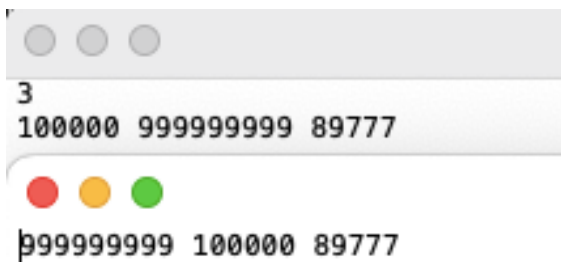
Результат работы кода на примерах из текста задачи:(скрины input output файлов):



```
6
31 41 59 26 41 58

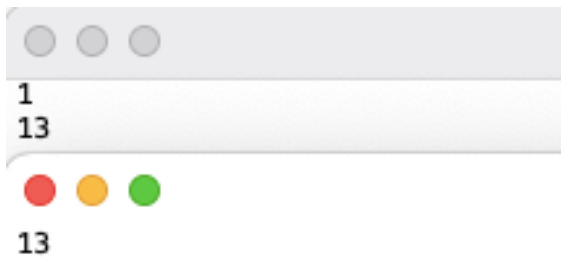
59 58 41 41 31 26
```

Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов):



```
3
100000 999999999 89777

999999999 100000 89777
```



```
1
13

13
```

Время выполнения:



```
Время работы: 0.0016263200086541474 секунд
```

Нижняя граница диапазона значений входных данных из текста задачи:

$1 \leq n$ (количество элементов)
 -10^{**9}

Верхняя граница диапазона значений входных данных из текста задачи:

$n \leq 10^3$ (количество элементов)
 10^9

Вывод по задаче:

Я научилась сортировать вставкой данные из файла по возрастанию и убыванию, а также преобразовала код из задач 1 и 4 с помощью метода swar.

Задача №4. Линейный поиск

Текст задачи:

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:

```
import time
t_start = time.perf_counter()

def len_search(second_line, first_line):
    ind = []
    count = 0
    for i in range(len(first_line)):
        if first_line[i] == second_line:
            ind.append(i)
            count+=1
    if count == 0:
        return('1')
    return(ind, count)

with open("input.txt", 'r') as file:
    first_line = file.readline().rstrip('\n')
    second_line = file.readline()
```

```

first_line = first_line.split()

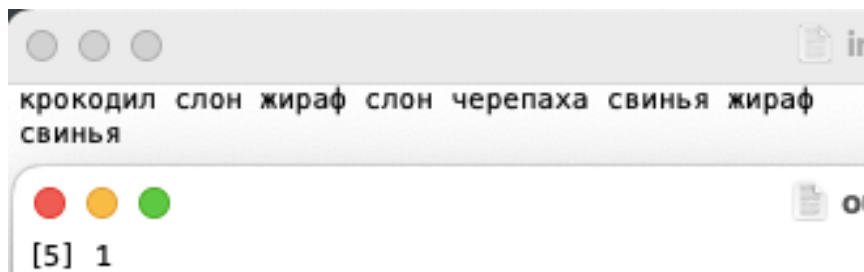
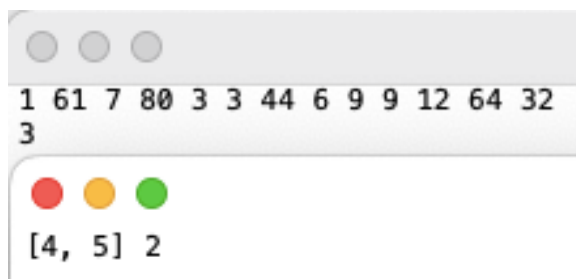
with open("output.txt", 'w') as file:
    a = len_search(second_line, first_line)
    a = [str(s) for s in a]
    file.write(' '.join(a))
print("Время работы: %s секунд" % (time.perf_counter()-
t_start))

```

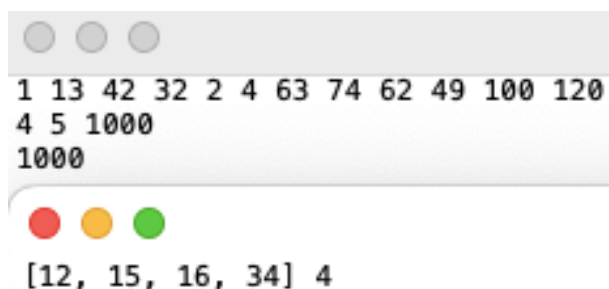
Текстовое объяснение решения:

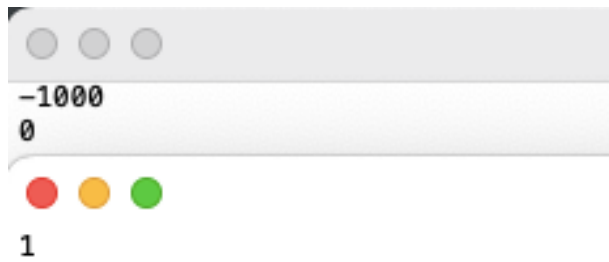
Я считываю данные из файла. Внутри функции проходим по всем значениям и ищем такое же значение как и во входном файле, а также считаем сколько значений найдено, если одинаковых значений нет, то выводим «1».

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов):





Время выполнения:

Время работы: 0.004228367994073778 секунд

Нижняя граница диапазона значений входных данных из текста задачи:

Ограничения: $0 \leq n \leq 10^3, -10^3 \leq a_i, V \leq 10^3$

Верхняя граница диапазона значений входных данных из текста задачи:

Ограничения: $0 \leq n \leq 10^3, -10^3 \leq a_i, V \leq 10^3$

Вывод по задаче:

Я научилась работать с файлами, составлять алгоритм линейного поиска по файлу с разными типами данных.

Задача №5. Сортировка выбором

Текст задачи:

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.

Листинг кода:

```
import time
t_start = time.perf_counter()

def sel_sort(second_line, first_line):
    n = int(first_line)
```

```

k = 0
while k != n-1:
    m = min(second_line[k:])
    ind = second_line.index(m)
    second_line[k], second_line[ind] =
second_line[ind], second_line[k]
    k+=1
return second_line

with open("input.txt", 'r') as file:
    first_line = file.readline().rstrip('\n')
    second_line = file.readline().split()
    second_line = [int(x) for x in second_line]

with open("output.txt", 'w') as file:
    a = sel_sort(second_line, first_line)
    a = [str(s) for s in a]
    file.write(' '.join(a))

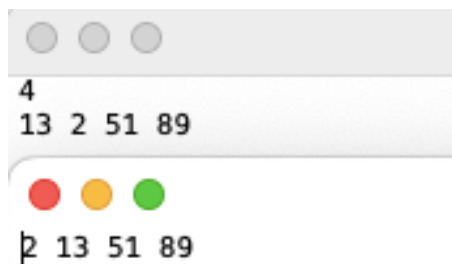
print("Время работы: %s секунд" % (time.perf_counter()-
t_start))

```

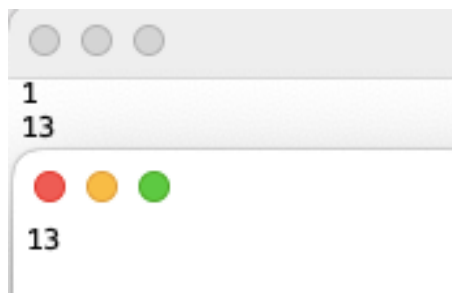
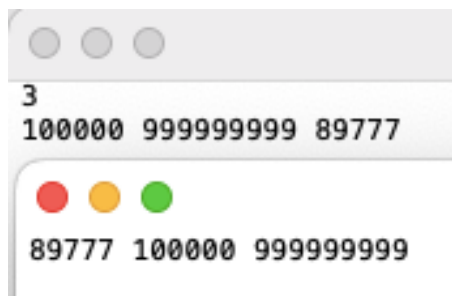
Текстовое объяснение решения:

Я считываю данные из файла. Внутри функции находим меньший элемент и ставим его на первое место, далее из оставшегося среза также находим наименьший элемент и ставим на следующее место.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов):



Время выполнения:

Время работы: 0.0039837539952714 секунд

Нижняя граница диапазона значений входных данных из текста задачи:

$1 \leq n$ (количество элементов)
 -10^{**9}

Верхняя граница диапазона значений входных данных из текста задачи:

$n \leq 10^{**3}$ (количество элементов)
 10^{**9}

Вывод по задаче:

Я научилась работать с файлами, искать наименьшее значение, брать срез, сортировать с помощью выбора.

Задача №10. Палиндром

Текст задачи:

Палиндром - это строка, которая читается одинаково как справа налево, так и слева направо.

На вход программы поступает набор больших латинских букв (не обязательно различных). Разрешается переставлять буквы, а также удалять некоторые буквы. Требуется из данных букв по указанным правилам составить палиндром наибольшей длины, а если таких палиндромов несколько, то выбрать первый из них в алфавитном порядке.

- **Формат входного файла (input.txt).** В первой строке входных данных содержится число n ($1 \leq n \leq 100000$). Во второй строке задается последовательность из n больших латинских букв (буквы записаны без пробелов).
- **Формат выходного файла (output.txt).** В единственной строке выходных данных выдайте искомым палиндром.

Листинг кода:

```
def palindrom_count (dict):
    count_l = {}
    for i in dict:
        if i in count_l:
            count_l[i] +=1
        else:
            count_l[i] = 1
    return count_l

def build_palindrom (dict):
    half = []
    middle = None
    for letter, count in dict.items():
        if count % 2 == 0:
            half.append(letter * (count // 2))
        else:
            half.append(letter * (count // 2))
            if middle is None:
                middle = letter
    half = ''.join(half)
    right_half = half[::-1]

    if middle:
        return half + middle + right_half
    else:
        return half + right_half

with open("input.txt", 'r') as file:
    first_line = file.readline().rstrip('\n')
    second_line = file.readline()
    second_line = sorted(second_line, key=lambda x:
```

```

x.lower())
    dict = ''.join(second_line)
    print(dict)

with open("output.txt", 'w') as file:
    a = palindrom_count(second_line)
    dict = palindrom_count(second_line)
    file.write(build_palindrom(dict))

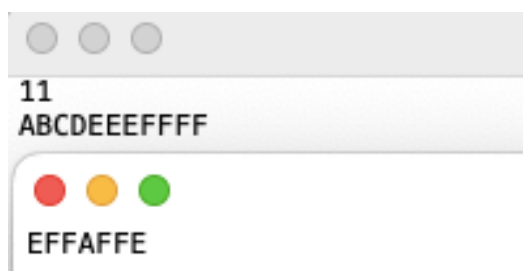
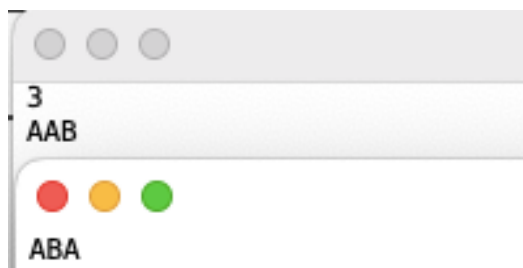
print(build_palindrom(dict))

```

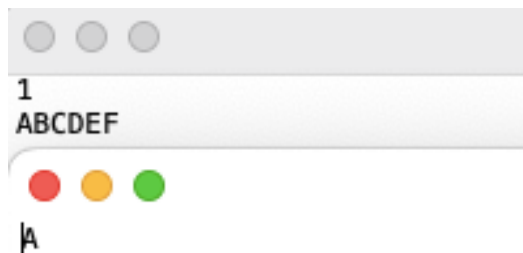
Текстовое объяснение решения:

Я считываю данные из файла. Далее сортируем строку по алфавиту и запускаем алгоритм. Внутри функции мы записываем в словарь букву и соответствующее ей числовое значение. Составляем из четного числа букв одну половину, если остается нечетный элемент добавляем его в середину и зеркально дублируем составленную часть палиндрома.

Результат работы кода на примерах из текста задачи:(скрины input output файлов):



Результат работы кода на максимальных и минимальных значениях:(скрины input output файлов):



Время выполнения:

Время работы: 0.002250971010653302 секунд

Нижняя граница диапазона значений входных данных из текста задачи:

1

Верхняя граница диапазона значений входных данных из текста задачи:

100000

Вывод по задаче:

Я научилась работать с файлами, с помощью использования словаря составлять палиндром, сортировать разные типы данных.

Вывод

При выполнении лабораторной я научилась базовым навыкам работы с кодом в Python: операциям над числами, вычислением времени работы кода, использовании функций и поиску решений по оптимизации алгоритмов.