САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «Алгоритмы и структуры данных»

Тема: Сортировка вставками, выбором, пузырьковая
Вариант 1

Выполнил:
Бен Шамех Абделазиз
Группа: K3239

Проверила:
Ромакина Оксана Михайловна

Санкт-Петербург
2025 г.

Содержание

1 Задача 1. Сортировка вставкой

1.1 Условие

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

1.2 Листинг кода

```
def task1():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    n = int(f.readline())
    arr = [int(i) for i in f.readline().split()]

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

    print(*ins_sort(arr), file=f2)
    f.close()
    f2.close()
```

1.3 Текстовое объяснение

Перебирая каждый элемент массива, начиная с индекса 1, ищем для него подходящее место в уже отсортированной части, переставляя элементы, которые больше него, "вправо пока не найдем корректную позицию.

1.4 Результаты выполнения

Пример из задачи:

- **Ввод (input.txt):** `6 \n 31 41 59 26 41 58`
- **Вывод (output.txt):** `26 31 41 41 58 59`

Метрики эффективности:

Сценарий	Время выполнения	Затраты памяти
Нижняя граница (n=1)	0.0065 sec	14.77 Mb
Пример из задачи	0.00407 sec	14.80 Mb
Верхняя граница (n=1000)	0.02734 sec	15.13 Mb

2 Задача 2. Сортировка вставкой +

2.1 Условие

Измените процедуру сортировки так, чтобы в выходном файле в первой строке отображалось n чисел — индексы, на которые перемещались элементы в момент обработки.

2.2 Листинг кода

```
def task2():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    n = int(f.readline())
    arr = [int(i) for i in f.readline().split()]

    def ins_sort(arr):
        ind_arr = [-1] * n
        ind_arr[0] = 1
        for j in range(1, len(arr)):
            i = j-1
            k = arr[j]
            while i >= 0 and arr[i] > k:
                arr[i+1] = arr[i]
                i = i-1
            arr[i+1] = k
            ind_arr[j] = i+2
        return ind_arr, arr

    arr1, arr2 = ins_sort(arr)
    print(*arr1, file=f2)
    print(*arr2, file=f2)
    f.close()
    f2.close()
```

2.3 Результаты выполнения

Ввод: `6 \n 31 41 59 26 41 58`
Вывод:
`1 2 3 1 4 5`
`26 31 41 41 58 59`

3 Задача 3. Сортировка вставкой по убыванию

3.1 Листинг кода

```python
def reverse_ins_sort(arr):
    for i in range(len(arr)-1):
        while i >= 0 and arr[i] < arr[i+1]:
            arr[i+1], arr[i] = arr[i], arr[i+1] # Swap procedure
            i -= 1
    return arr
```

3.2 Текстовое объяснение

Выполняем операции, аналогичные обычной сортировке вставками, но вместо сдвига используем процедуру `swap` (обмен местами), и знак сравнения меняется на противоположный для достижения невозрастающего порядка.

3.3 Результаты выполнения

Ввод: `6 \n 31 41 59 26 41 58` → **Вывод:** `59 58 41 41 31 26`.

4 Задача 4. Линейный поиск

4.1 Условие

Реализовать линейный поиск значения V. Если число встречается несколько раз, вывести количество вхождений и все индексы через запятую.

4.2 Листинг кода

```
def task4():
    f = open('input.txt')
    arr = [int(i) for i in f.readline().split()]
    v = int(f.readline())
    found = [i for i, x in enumerate(arr) if x == v]

    if not found:
        print(-1)
    else:
        print(len(found))
        print(*(found))
```

4.3 Результаты выполнения

Ввод (V=41): `31 41 59 26 41 58 \n 41` → **Вывод:** `2 \n 1 4`.

5 Задача 7. Знакомство с жителями Сортлэнда

5.1 Условие

Найти идентификационные номера самого бедного, среднего и самого богатого жителей Сортлэнда.

5.2 Листинг кода

```
def task7():
    f = open('input.txt')
    n = int(f.readline())
    arr = [(float(item), ind+1) for ind, item in enumerate(f.readline()
   .split())]
    # Insertion sort for tuples (money, id)
    for j in range(1, len(arr)):
        i = j - 1; k = arr[j]
        while i >= 0 and arr[i][0] > k[0]:
            arr[i+1] = arr[i]; i -= 1
        arr[i+1] = k
    print(arr[0][1], arr[n//2][1], arr[-1][1])
```

5.3 Результаты выполнения

Ввод: `5 \n 10.1 55.0 40.0 20.0 70.5` → **Вывод:** `1 3 5`.

6 Задача 10. Палиндром

6.1 Листинг кода

```python
def solve_palindrome(s):
    from collections import Counter
    cnt = Counter(s)
    half = ""; mid = ""
    for char in sorted(cnt.keys()):
        if cnt[char] % 2 != 0 and mid == "":
            mid = char
        half += char * (cnt[char] // 2)
    return half + mid + half[::-1]
```

6.2 Результаты выполнения

Ввод: `7 \n AABCDBA` → **Вывод:** `ABCDCBA`.

7 Вывод

В ходе лабораторной работы были освоены алгоритмы сортировки со сложностью $O(n^2)$ (вставками) и их модификации. Было выявлено, что:

1. Алгоритм сортировки вставками эффективен для массивов небольшого размера.
2. Временная сложность в худшем случае составляет $O(n^2)$, а затраты памяти — $O(1)$ (In-place), что подтверждено метриками.
3. Линейный поиск эффективен при $n \leq 10^3$, но при росте данных требует оптимизации до бинарного поиска.