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 14

Выполнил:

Савченко В. А.

К3139

Проверил:

Афанасьев А. В.

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc178011417)

[Задачи по варианту 3](#_Toc178011418)

[Задача №1. Ввод-вывод 3](#_Toc178011419)

[Задача №1.1 3](#_Toc178011420)

[Задача №1.2 4](#_Toc178011421)

[Задача №1.3 5](#_Toc178011422)

[Задача №1.4 6](#_Toc178011423)

[Задача №2. Число Фибоначчи 7](#_Toc178011424)

[Задача №3. Еще про числа Фибоначчи 10](#_Toc178011425)

[Вывод 13](#_Toc178011426)

# Задачи по варианту

## Задача №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 ≤ n ≤ 103 ) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 109 .

Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.

Код:

import time  
t\_start = time.perf\_counter()  
  
def insertion\_sort(arr):  
 n = len(arr)  
 for i in range(1, n):  
 key = arr[i]  
 j = i - 1  
  
 while j >= 0 and arr[j] > key:  
 arr[j + 1] = arr[j]  
 j -= 1  
 arr[j + 1] = key  
  
  
with open('input.txt', 'r') as file:  
 n = int(file.readline().strip())  
 arr = list(map(int, file.readline().strip().split()))  
  
insertion\_sort(arr)  
  
with open('output.txt', 'w') as file:  
 file.write(' '.join(map(str, arr)))  
  
print(time.perf\_counter() - t\_start, 'секунд')

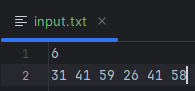
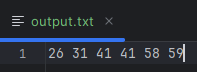
Текстовое объяснение решения:

В функции insertion\_sort() цикл проходит по массиву и для каждого элемента (начиная со второго) находит правильную позицию для него в отсортированной части массива

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

## Задача №3. Сортировка вставкой по убыванию

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.

Код:

import time  
t\_start = time.perf\_counter()  
  
def swap(arr, i, j):  
 arr[i], arr[j] = arr[j], arr[i]  
  
def insertion\_sort(arr):  
 n = len(arr)  
 for i in range(1, n):  
 key = arr[i]  
 j = i - 1  
  
 while j >= 0 and arr[j] < key:  
 swap(arr, j, j + 1)  
 j -= 1  
 arr[j + 1] = key  
  
  
with open('input.txt', 'r') as file:  
 n = int(file.readline().strip())  
 arr = list(map(int, file.readline().strip().split()))  
  
insertion\_sort(arr)  
  
with open('output.txt', 'w') as file:  
 file.write(' '.join(map(str, arr)))  
  
print(time.perf\_counter() - t\_start, 'секунд')

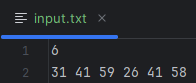
Текстовое объяснение решения:

В функции insertion\_sort() цикл проходит по массиву и для каждого элемента (начиная со второго) находит правильную позицию для него в отсортированной части массива. В отличии от первой задачи, сортировка проходит по убыванию и используется функция swap

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

## Задача №4. Линейный поиск

Рассмотрим задачу поиска. • Формат входного файла. Последовательность изn чисел A = a1, a2, . . . , an в первой строке, числа разделены пробелом, и значение V во второй строке. Ограничения: 0 ≤ n ≤ 103 , −103 ≤ ai , V ≤ 103 • 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.

Код:

import time  
t\_start = time.perf\_counter()  
  
def linear\_search(arr, v):  
 for i in range(len(arr)):  
 if arr[i] == v:  
 return i  
 return -1  
  
  
with open('input.txt', 'r') as file:  
 arr = file.readline().strip().split()  
 v = file.readline().strip()  
  
 if arr[0].isdigit():  
 arr = list(map(int, arr))  
 v = int(v)  
  
result = linear\_search(arr, v)  
  
with open('output.txt', 'w') as file:  
 file.write(str(result))  
  
print(time.perf\_counter() - t\_start, 'секунд')

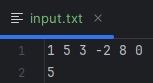
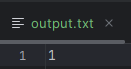
Текстовое объяснение решения:

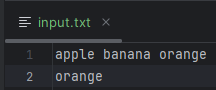
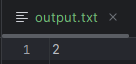
Функция linear\_search() выполняет линейный поиск по массиву arr, сравнивая каждый элемент с искомым значением v.

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

## Задача №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.

Код:

import time  
t\_start = time.perf\_counter()  
  
def selection\_sort(arr):  
 n = len(arr)  
 for i in range(n):  
 min\_index = i  
 for j in range(i + 1, n):  
 if arr[j] < arr[min\_index]:  
 min\_index = j  
 arr[i], arr[min\_index] = arr[min\_index], arr[i]  
  
  
with open('input.txt', 'r') as file:  
 n = int(file.readline().strip())  
 arr = list(map(int, file.readline().strip().split()))  
  
selection\_sort(arr)  
  
with open('output.txt', 'w') as file:  
 file.write(' '.join(map(str, arr)))  
  
print(time.perf\_counter() - t\_start, 'секунд')

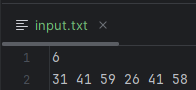
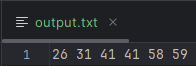
Текстовое объяснение решения:

Функция находит наименьший элемент из неотсортированной части массива и перемещая его в начало. Этот процесс повторяется для оставшейся части массива до тех пор, пока весь массив не будет отсортирован.

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

## Задача №6. Сортировка выбором

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректоности процедуры вам необходимо доказать, что она завершается и что A′ [1] ≤ A′ [2] ≤ ... ≤ A′ [n], где A′ - выход процедуры Bubble\_Sort, a 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.

Код:

import time  
t\_start = time.perf\_counter()  
  
def bubble\_sort(arr):  
 n = len(arr)  
 for i in range(n - 1):  
 for j in range(n - 1, i, -1):  
 if arr[j] < arr[j - 1]:  
 arr[j], arr[j - 1] = arr[j - 1], arr[j]  
  
  
with open('input.txt', 'r') as file:  
 n = int(file.readline().strip())  
 arr = list(map(int, file.readline().strip().split()))  
  
bubble\_sort(arr)  
  
with open('output.txt', 'w') as file:  
 file.write(' '.join(map(str, arr)))  
  
print(time.perf\_counter() - t\_start, 'секунд')

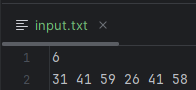
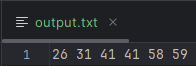
Текстовое объяснение решения:

Функция проходит по массиву несколько раз, сравнивая и переставляя соседние элементы, если они находятся в неправильном порядке.

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

# Вывод

В данной лабораторной работе я познакомился с различными методами сортировки и реализовал их. Для этого мне потребовалось работать с массивами и циклами