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: Лабораторная работа 1.

Вариант 21

Выполнил:

Ступичев М. Н.

К3139

Проверила:

…

Санкт-Петербург

2024 г.

# Содержание отчета

[Содержание отчета 2](#_Toc183359051)

[Задачи по варианту 3](#_Toc183359052)

[Задача №1. Сортировка вставкой. 3](#_Toc183359053)

[Задача №5. Сортировка выбором. 4](#_Toc183359054)

[Задача №6. Пузырьковая сортировка. 5](#_Toc183359055)

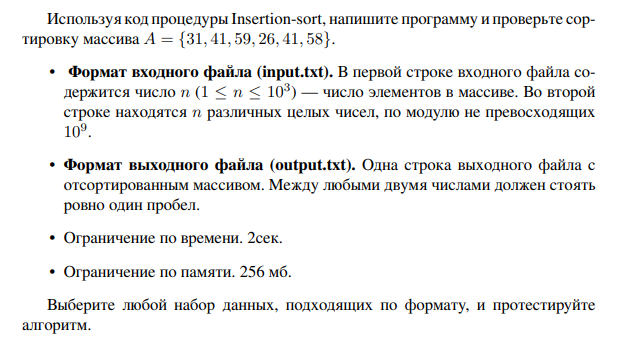
[Задача №9. Сложение двоичных чисел. 6](#_Toc183359056)

[Задача №10. Палиндром. 7](#_Toc183359057)

[Вывод 8](#_Toc183359058)

# Задачи по варианту

## Задача №1. Сортировка вставкой.



Решение:

def insertion\_sort(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

​

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task1/tests/input', 'r') as f:

   n = int(f.readline())

   arr = list(map(int, f.readline().split()))

​

insertion\_sort(arr)

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task1/tests/output', 'w') as f:

   f.write(' '.join(map(str, arr)))

​

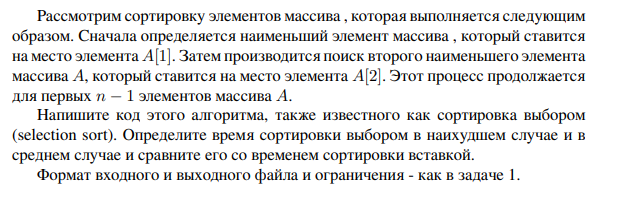
Текстовое объяснение решения.

Алгоритм сортировки вставкой работает итеративно, проходя по массиву справа налево. На каждой итерации, текущий элемент рассматривается как ключ. Затем алгоритм сравнивает ключ с элементами, расположенными слева от него, до тех пор, пока не найдет элемент меньше или равный ключу, или пока не достигнет начала массива. Если найден элемент меньше ключа, ключ вставляется на место после этого элемента, а все элементы между найденным элементом и ключом сдвигаются на одну позицию вправо. В противном случае, ключ вставляется в начало массива. Этот процесс повторяется для каждого элемента, эффективно создавая отсортированный массив. Алгоритм эффективен для небольших массивов и уже частично отсортированных данных.

Вывод по задаче:

Реализован стандартный алгоритм сортировки вставкой.

## Задача №5. Сортировка выбором.



Решение:

def selection\_sort(arr):

   for i in range(len(arr)):

       min\_index = i

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

           if arr[j] < arr[min\_index]:

               min\_index = j

​

       arr[i], arr[min\_index] = arr[min\_index], arr[i]

​

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task5/tests/input', 'r') as f:

   n = int(f.readline())

   arr = list(map(int, f.readline().split()))

​

selection\_sort(arr)

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task5/tests/output', 'w') as f:

   f.write(' '.join(map(str, arr)))

​

​

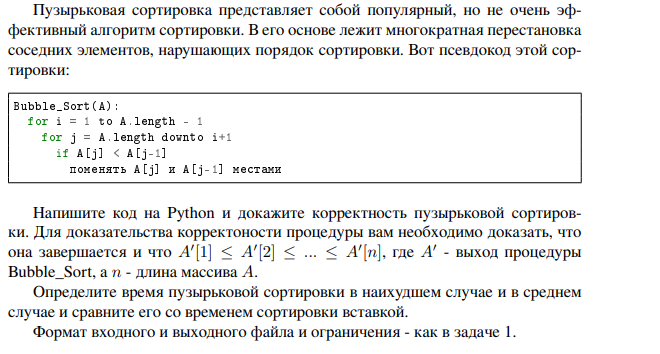
Текстовое объяснение решения.

Этот код реализует сортировку выбором. Сначала он читает из файла input количество элементов n и сам массив. Затем функция selection\_sort проходит по массиву. На каждой итерации внешнего цикла она находит минимальный элемент в оставшейся неотсортированной части массива и меняет его местами с элементом, находящимся в начале неотсортированной части. Внутренний цикл отвечает за поиск минимального элемента. После завершения внешнего цикла массив оказывается отсортированным. Наконец, отсортированный массив записывается в файл output.

Вывод по задаче:

Реализована сортировка вставкой.

## Задача №6. Пузырьковая сортировка.



Решение:

def bubble\_sort(data):

   n = len(data)

   for i in range(n - 1):

       for j in range(n - i - 1):

           if data[j] > data[j + 1]:

               data[j], data[j + 1] = data[j + 1], data[j]

​

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task6/tests/input', 'r') as f:

   n = int(f.readline())

   arr = list(map(int, f.readline().split()))

​

bubble\_sort(arr)

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task6/tests/output', 'w') as f:

   f.write(' '.join(map(str, arr)))

​

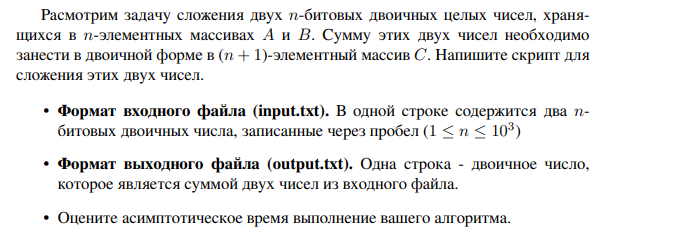
Текстовое объяснение решения.

Этот код реализует алгоритм сортировки пузырьком. Сначала он читает из файла input размер массива и сам массив. Функция bubble\_sort использует два вложенных цикла. Внешний цикл проходит по массиву столько раз, сколько элементов в нём минус один. Внутренний цикл сравнивает соседние элементы и меняет их местами, если они находятся в неправильном порядке (элемент слева больше элемента справа). После каждого прохода внешнего цикла, наибольший неотсортированный элемент “всплывает” на свое правильное место в конце неотсортированной части массива. В результате многократных проходов, массив становится полностью отсортированным. Наконец, отсортированный массив записывается в файл output.

Вывод по задаче:

Реализована сортировка пузырьком.

## Задача №9. Сложение двоичных чисел.



Решение:

def binary\_add(A, B):

   n = len(A)

   C = [0] \* (n + 1)

   extra = 0

​

   for i in range(n - 1, -1, -1):

       total = A[i] + B[i] + extra

       C[i + 1] = total % 2

       extra = total // 2

​

   C[0] = extra

   return C

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task9/tests/input', 'r') as f:

   line = f.readline()

   A\_str, B\_str = line.split()

   A = list(map(int, A\_str))

   B = list(map(int, B\_str))

​

C = binary\_add(A, B)

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task9/tests/output', 'w') as f:

   f.write(''.join(map(str, C)))

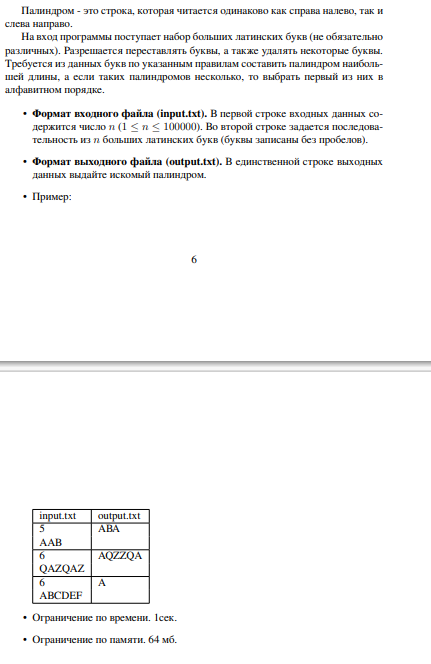
Текстовое объяснение решения.

Этот код выполняет суммирование двух двоичных чисел, представленных в виде массивов целых чисел (0 и 1). Он считывает два двоичных числа A и B из входного файла построчно. Функция binary\_add суммирует числа поразрядно, начиная с младшего разряда. Переменная extra хранит перенос из предыдущего разряда. На каждой итерации, складываются соответствующие биты A, B и перенос, результат по модулю 2 записывается в результирующий массив C (этот бит и есть результат сложения), а целочисленное деление на 2 даёт новый перенос. После обработки всех разрядов, оставшийся перенос записывается в старший разряд C. Наконец, результирующее двоичное число (массив C) записывается в выходной файл.

Вывод по задаче:

Реализован алгоритм сложения двух двоичных чисел.

## Задача №10. Палиндром.



Решение:

def get\_pallindrom(string):

   letters = set([let for let in string])

   count\_letters = dict()

   for letter in letters:

       count\_letters[letter] = string.count(letter)

​

   first\_part = ""

   possible\_middle\_letter = []

​

   for let, count in count\_letters.items():

       if count >= 2:

           first\_part += let \* (count // 2)

           if count % 2 == 1:

               possible\_middle\_letter.append(let)

       if count == 1:

           possible\_middle\_letter.append(let)

​

   if possible\_middle\_letter:

       possible\_middle\_letter.sort()

       middle\_letter = possible\_middle\_letter[0]

   else:

       middle\_letter = ''

​

   first\_part = ''.join(sorted([let for let in first\_part]))

   second\_part = first\_part[::-1]

​

   return first\_part + middle\_letter + second\_part

​

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task10/tests/input', 'r') as f:

   n = int(f.readline())

   string = f.readline().strip()

​

string = get\_pallindrom(string)

​

with open('C:/Users/Михаил/PycharmProjects/algorithm\_and\_data\_structures/lab1/task10/tests/output', 'w') as f:

   f.write(string)

​

Текстовое объяснение решения.

Этот код строит максимально длинный палиндром из заданной строки. Сначала он подсчитывает количество вхождений каждого символа. Затем формирует первую половину палиндрома (first\_part), используя каждый символ столько раз, сколько позволяет его чётное количество вхождений. Символы с нечётным количеством вхождений добавляются в список possible\_middle\_letter. Если такой список не пуст, то символ с наименьшим кодом из этого списка используется в качестве центральной буквы палиндрома (middle\_letter). Вторая половина палиндрома (second\_part) создаётся как обращение первой половины. В итоге, функция возвращает конкатенацию первой половины, центральной буквы (если есть) и второй половины, что и составляет полученный палиндром. Код также включает чтение строки из входного файла и запись результата в выходной файл.

Вывод по задаче:

Реализован алгоритм нахождения максимально возможного палиндрома.

# Вывод

Решены 3 задачи из 21 варианта и плюсом 6 и 10.