(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 10

Выполнил:

Кретов И.А.

К3139

Проверил:

Афанасьев А.В

Санкт-Петербург

2024 г.

# Содержание отчета

[**Содержание отчета**](#_heading=h.gjdgxs) **2**

[**Задачи по варианту**](#_heading=h.30j0zll) **3**

[Задача №1.](#_heading=h.1fob9te) Сортировка вставкой 3

[Задача №3.](#_heading=h.1fob9te) Сортировка вставкой по убыванию 5

[Задача №7. Знакомство с жителями Сортлэнда](#_heading=h.1fob9te)  7

[**Дополнительные задачи**](#_heading=h.3znysh7) **10**

[Задача №4.](#_heading=h.2et92p0) Линейный поиск 10

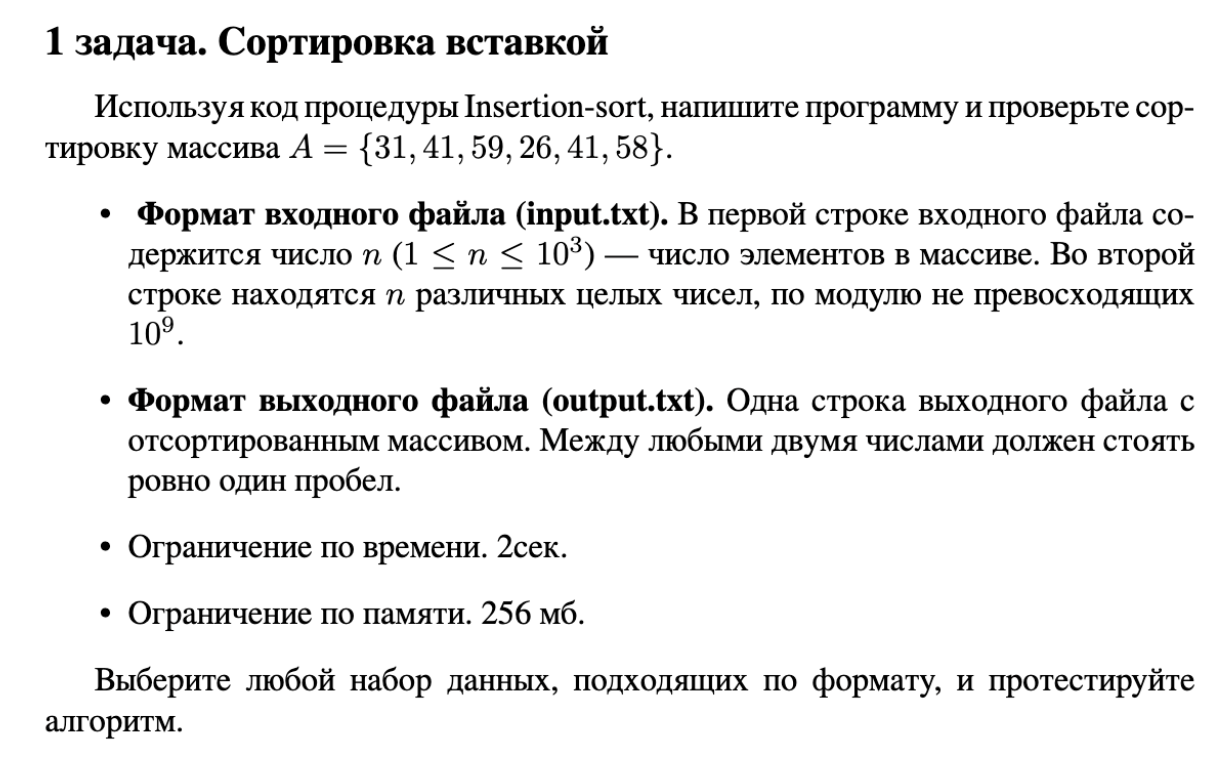
[Задача №6. Пузырьковая сортировка](#_heading=h.1fob9te) 12

[Задача №](#_heading=h.1fob9te)9. Сложение двоичных чисел 14

[**Вывод**](#_heading=h.tyjcwt) **16**

# Задачи по варианту

## Задача №1. Сортировка вставкой



import time

import tracemalloc

tracemalloc.start()

t\_start = time.perf\_counter()

f = open("input.txt", "r")

n = int(f.readline())

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

f.close()

for i in range(1, n):

j = i-1

temp = a[i]

while j >= 0 and temp < a[j]:

a[j+1] = a[j]

j -= 1

a[j+1] = temp

f = open("output.txt", "w")

for i in range(n):

f.write(str(a[i])+" ")

f.close()

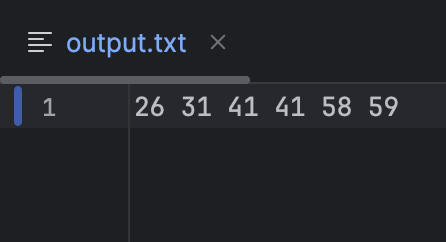
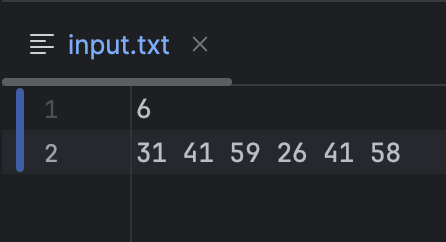
print("Время работы: %s секунд " % (time.perf\_counter() - t\_start))

print("Память:", tracemalloc.get\_traced\_memory()[1] / (1024 \*\* 2), "МБ" )

tracemalloc.stop()

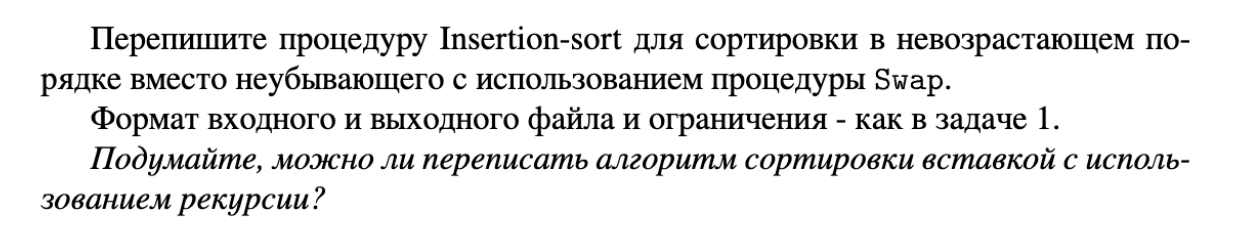
Текстовое объяснение решения:

1. Открываем файл и читаем оттуда массив
2. Записываем a[i] в временную переменную. Для каждого элемента (начиная со i-1), программа сравнивает его с предыдущими и сдвигает элементы, пока не найдет элемент меньше temp.
3. Открываем файл и записываем результат сортировки массива



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.00027004198636859655 секунд | 0.013161659240722656 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.1894567499984987 секунд | 0.09668445587158203 МБ |

## Задача №3. Сортировка вставкой по убыванию



import time

import tracemalloc

tracemalloc.start()

t\_start = time.perf\_counter()

f = open("input.txt", "r")

n = int(f.readline())

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

f.close()

for i in range(1, n):

j = i-1

temp = a[i]

while j >= 0 and temp > a[j]:

a[j+1] = a[j]

j -= 1

a[j+1] = temp

f = open("output.txt", "w")

for i in range(n):

f.write(str(a[i])+" ")

f.close()

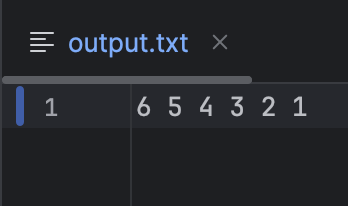
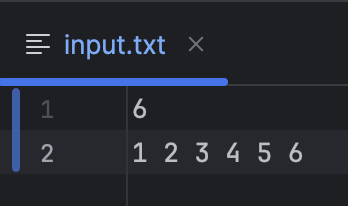
print("Время работы: %s секунд " % (time.perf\_counter() - t\_start))

print("Память:", tracemalloc.get\_traced\_memory()[1] / (1024 \*\* 2), "МБ" )

tracemalloc.stop()

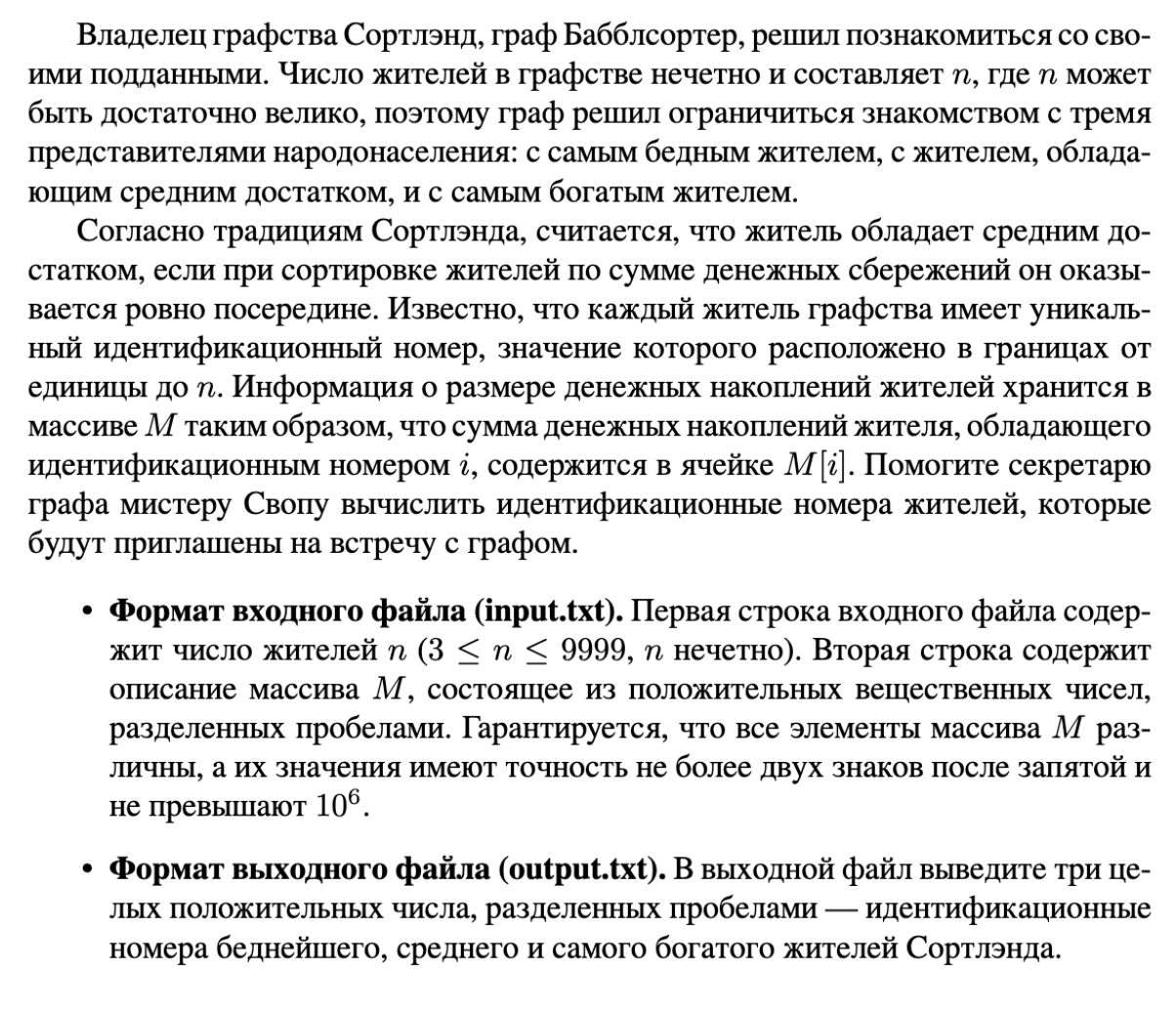
Текстовое объяснение решения:

1. Аналогично предыдущей задачи, но мы сдвигаем элементы, пока не найдем элемент, который больше temp



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.000292125012492761 секунд | 0.013161659240722656 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.19306316698202863 секунд | 0.09668445587158203 МБ |

## Задача №7. Знакомство с жителями Сортлэнда



import time

import tracemalloc

tracemalloc.start()

t\_start = time.perf\_counter()

f = open("input.txt", "r")

n = int(f.readline())

a = list(map(float,f.readline().split()))

f.close()

b = a.copy()

for i in range(n):

for j in range(n):

if a[i] < a[j]:

a[i], a[j] = a[j], a[i]

s = ["-1", "-1", "-1"]

for i in range(n):

if b[i] == a[0]:

s[0] = str(i+1)

elif b[i] == a[n//2]:

s[1] = str(i+1)

elif b[i] == a[-1]:

s[2] = str(i+1)

f = open("output.txt", "w")

for i in range(3):

f.write(s[i]+" ")

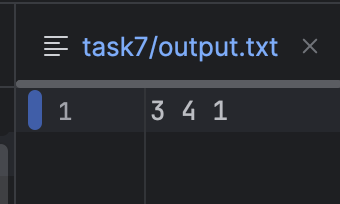
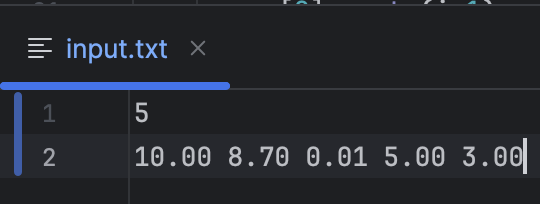
f.close()

print("Время работы: %s секунд " % (time.perf\_counter() - t\_start))

print("Память:", tracemalloc.get\_traced\_memory()[1] / (1024 \*\* 2), "МБ" )

tracemalloc.stop()

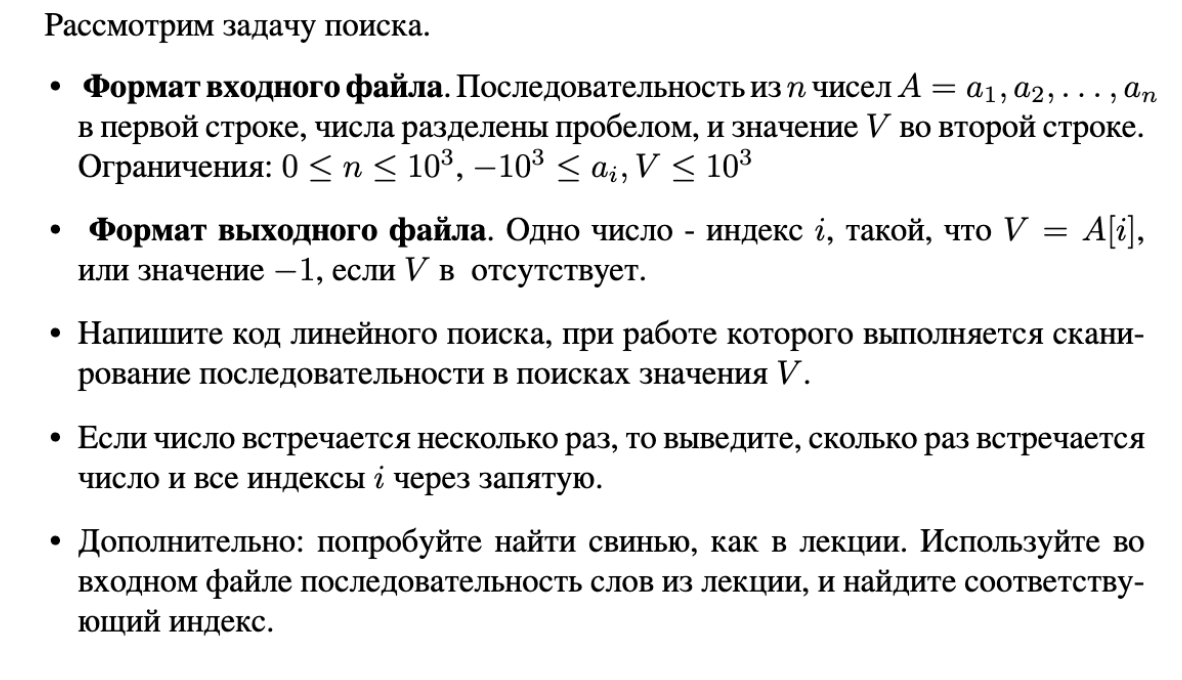
1. Открываем файл и читаем оттуда массив “a”
2. Копируем массив “a” в массив “b”
3. Сортируем массив “a” при помощи пузырьковой сортировки
4. При помощи линейного поиска ищем в массиве “b” элементы равные a[0], a[n//2] и a[-1] и записываем их индексы
5. Записываем в файл индексы, которые мы записали до этого



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0003218750061932951 секунд | 0.013228416442871094 МБ |
| Пример из задачи | 0.0006273750041145831 секунд | 0.013247489929199219 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.3391752500028815 секунд | 0.09265518188476562 МБ |

# Дополнительные задачи

## Задача №4. Линейный поиск



import time

import tracemalloc

tracemalloc.start()

t\_start = time.perf\_counter()

f = open("input.txt", "r")

a = list(f.readline().split())

v = f.readline()

f.close()

k = []

for i in range(len(a)):

if a[i] == v:

k.append(i)

f = open("output.txt", "w")

if len(k) == 0:

f.write("-1")

elif len(k) == 1:

f.write(str(k[0]))

else:

f.write(str(len(k))+"\n")

for i in range(len(k)-1):

f.write(str(k[i])+',')

f.write(str(k[len(k)-1]))

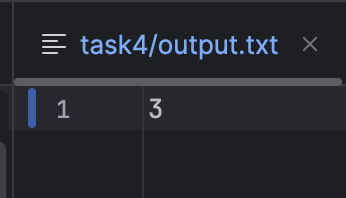
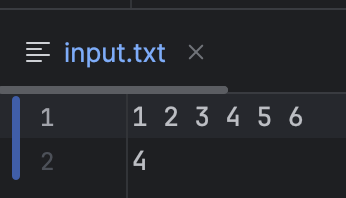
f.close()

print("Время работы: %s секунд " % (time.perf\_counter() - t\_start))

print("Память:", tracemalloc.get\_traced\_memory()[1] / (1024 \*\* 2), "МБ" )

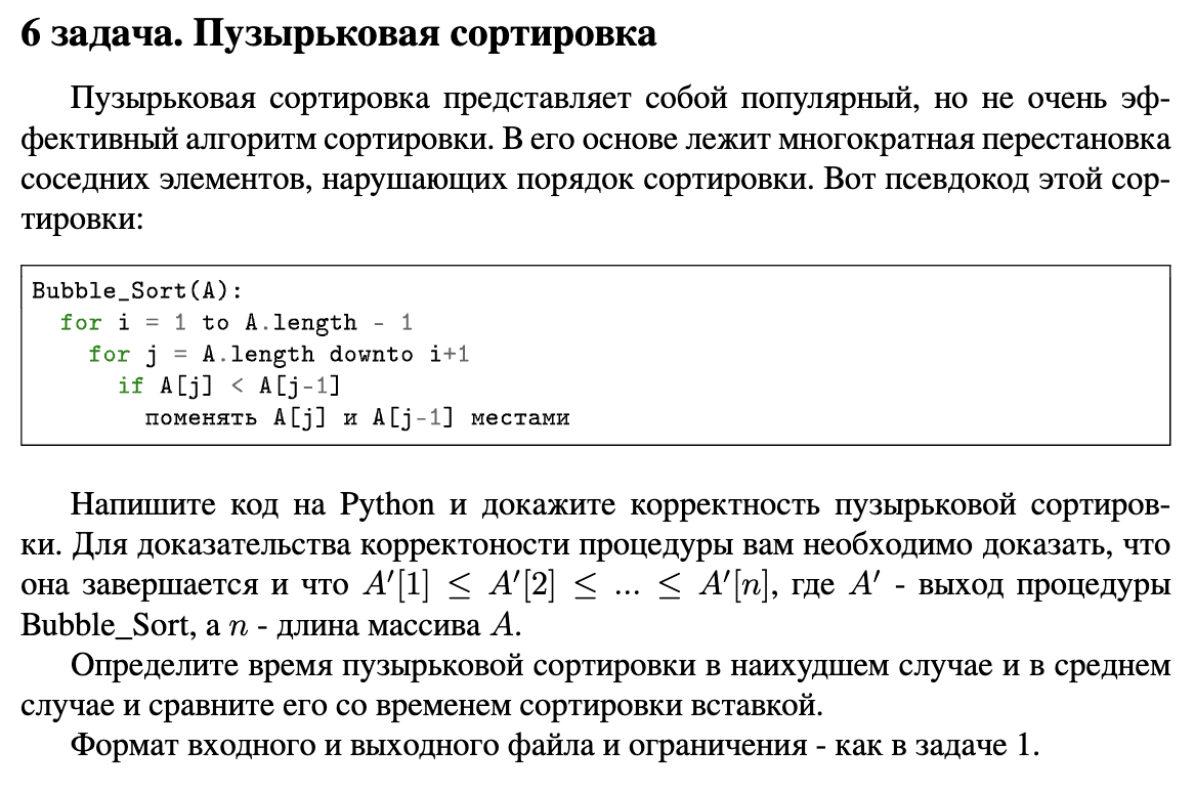
tracemalloc.stop()

1. Открываем файл и читаем оттуда массив
2. В цикле проверяем каждый элемент если он равен v, то записываем его в массив k
3. Записываем массив k в файл



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0002422079851385206 секунд | 0.013165473937988281 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0008749580010771751 секунд | 0.07173347473144531 МБ |

## **Задача №**6**.** Пузырьковая сортировка



import time

import tracemalloc

tracemalloc.start()

t\_start = time.perf\_counter()

f = open("input.txt", "r")

n = int(f.readline())

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

f.close()

for i in range(n):

for j in range(n-1,i,-1):

if a[j] < a[j-1]:

a[j], a[j-1] = a[j-1], a[j]

f = open("output.txt", "w")

for i in range(n):

f.write(str(a[i]) + " ")

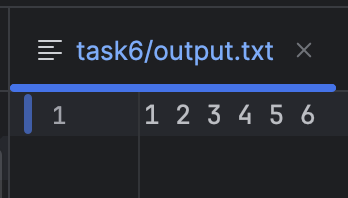
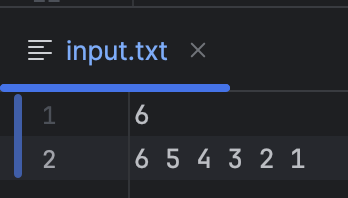
f.close()

print("Время работы: %s секунд " % (time.perf\_counter() - t\_start))

print("Память:", tracemalloc.get\_traced\_memory()[1] / (1024 \*\* 2), "МБ" )

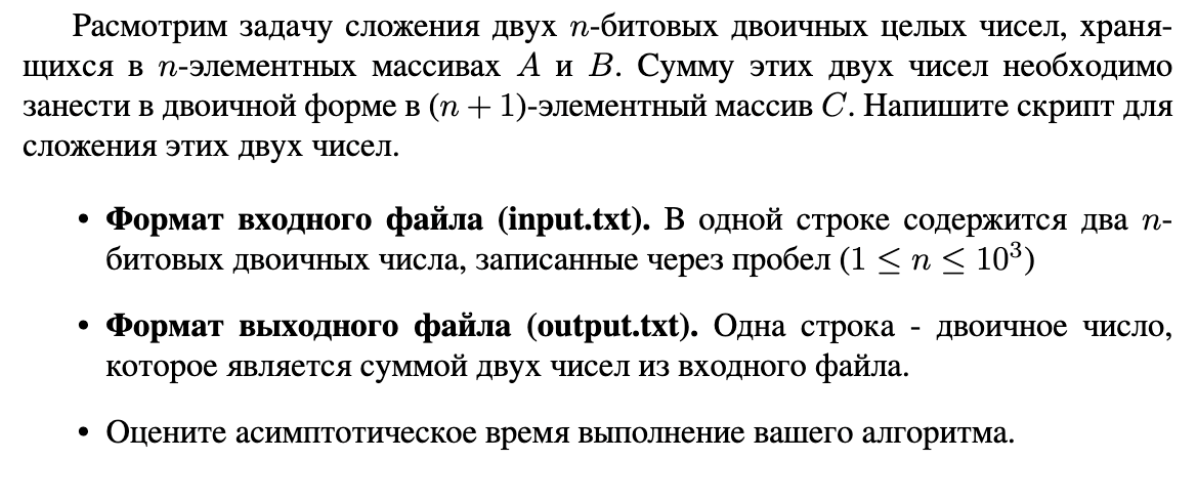
tracemalloc.stop()

1. Открываем файл и читаем оттуда массив
2. В цикле последовательно сравниваем пары соседних элементов в массиве и меняем их местами
3. Записываем в файл результат работы алгоритма



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0002376670017838478 секунд | 0.013161659240722656 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.6093550000223331 секунд | 0.09212684631347656 МБ |

## Задача №9. Сложение двоичный чисел



import time

import tracemalloc

tracemalloc.start()

t\_start = time.perf\_counter()

f = open("input.txt", "r")

s1, s2 = f.readline().split()

f.close()

s3 = [0]

for i in range(len(s1)):

s3.append(int(s1[i]) + int(s2[i]))

for i in range(1,len(s3)):

for j in range(i, len(s3)):

if int(s3[j]) == 2:

s3[j-1] += 1

s3[j] = 0

f = open("output.txt", "w")

for i in range(len(s3)):

f.write(str(s3[i]))

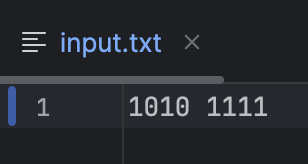
f.close()

print("Время работы: %s секунд " % (time.perf\_counter() - t\_start))

print("Память:", tracemalloc.get\_traced\_memory()[1] / (1024 \*\* 2), "МБ" )

tracemalloc.stop()

1. Открываем файл и читаем оттуда два числа
2. В новый массив записываем суммы сложения i элементов чисел
3. В цикле проверяем если элемент нового массива равен 2, то меняем его на 0, а элемент до него увеличиваем на 1
4. Записываем в файл получившееся двоичное число



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0002455420035403222 секунд | 0.013203620910644531 МБ |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.26088179100770503 секунд | 0.06496524810791016 МБ |

# 

# Вывод

В результате выполнения данной лабораторной работы было изучен алгоритм сортировки вставкой, пузырьковой сортировки, линейного поиска и сложения двоичных чисел