САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2
по курсу «Алгоритмы и структуры данных»
Тема: Сортировка слиянием. Метод декомпозиции

Выполнил:
Волжева М.И.
К3141

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

Содержание отчета

Задачи по варианту

Задача №1. Сортировка слиянием

Текст задачи:

- Используя псевдокод процедур Merge и Merge-sort из презентации к Лекции 2 (страницы 6-7), напишите программу сортировки слиянием на Python и проверьте сортировку, создав несколько рандомных массивов, подходящих под параметры:
- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 2 \cdot 10^4$) — 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.
- Пример:

Input.txt	10 45 -8 -61 97 77 44 -66 -19 84 15
Output.txt	-66 -61 -19 -8 15 44 45 77 84 97

Листинг кода:

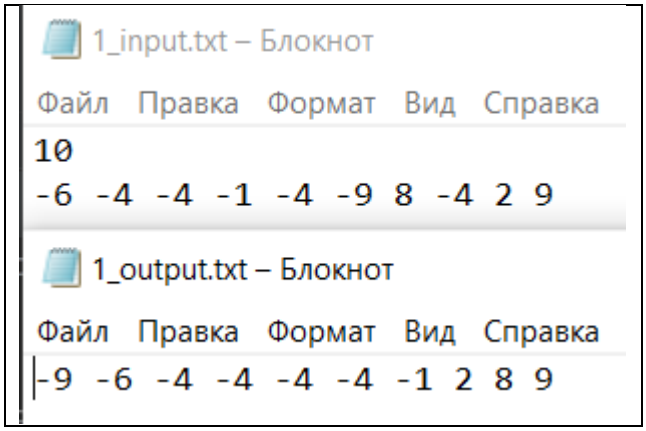
```
import time
import os, psutil
import random


def merge(list1, list2):
    list3 = []
    i = j = 0
    while i < len(list1) or j < len(list2):
        if j == len(list2) or (i < len(list1) and list1[i] < list2[j]):
            list3.append(list1[i])
            i += 1
        else:
            list3.append(list2[j])
            j += 1
    return list3


def merge_sort(arr):
    if len(arr) == 1:
        return arr
```

```
    left = merge_sort(arr[: len(arr) // 2])
    right = merge_sort(arr[len(arr) // 2:])

    return merge(left, right)


def test(a, b):
    k = []
    for i in range(0, a):
        k.append(random.randint(-b, b))
    return k


def test_test(arr):
    k = False
    for i in range(0, len(arr)-1):
        if arr[i] <= arr[i+1]:
            return False
    return True


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("1_input.txt", "w")
m = open("1_output.txt", "w")

f.write("10\n")
numbers = test(10, 100)
f.write(" ".join(map(str, numbers)))

sorted_numbers = merge_sort(numbers)
if test_test(sorted_numbers):
    m.write(" ".join(map(str, merge_sort(numbers))))

f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Я переписала псевдокод данный в презентации, далее написала функцию для генерации массива и функцию для проверки отсортированного массива.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001947299999999999 second	13.703125 mb
Пример из задачи	0.002422599999999997 second	13.61328125 mb
Верхняя граница диапазона значений входных данных из текста задачи	1.0784858 second	19.48828125 mb

Анализ времени в наилучшем и наихудшем случае:
Наилучший случай – массив отсортирован, наихудший случай - массив отсортирован в обратном порядке. Для тестов был использован массив длиной 10 элементов.

	Время в наилучшем случае	Время в наихудшем случае
merge_sort	0.0019613999999999988 second	0.0018620000000000025 second
insertion_sort	0.003251000000000004 second	0.005275799999999969 second

Вывод по задаче:.Мы научились писать сортировку слияние и убедились, что она работает быстрее других, ранее изученных сортировок.

Задача №2. Сортировка слиянием+

Текст задачи:

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью сортировки слиянием. Чтобы убедиться, что Вы действительно используете сортировку слиянием, мы просим Вас, после каждого осуществленного слияния (то есть, когда соответствующий подмассив уже отсортирован!), выводить индексы граничных элементов и их значения.

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^5$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- Формат выходного файла (output.txt). Выходной файл состоит из нескольких строк. 2 – В последней строке выходного файла требуется вывести отсортированный в порядке неубывания массив, данный на входе. Между любыми двумя числами должен стоять ровно один пробел. Все предшествующие строки описывают осуществленные слияния, по одному на каждой строке. Каждая такая строка должна содержать по четыре числа: If , Il , Vf , Vl , где If — индекс начала области слияния, Il — индекс конца области слияния, Vf — значение первого элемента области слияния, Vl — значение последнего элемента области слияния. – Все индексы начинаются с единицы (то есть, $1 \le If \le Il \le n$). Индексы области слияния должны описывать положение области слияния в исходном массиве! Допускается не выводить информацию о слиянии для подмассива длиной 1, так как он отсортирован по определению.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

Листинг кода:

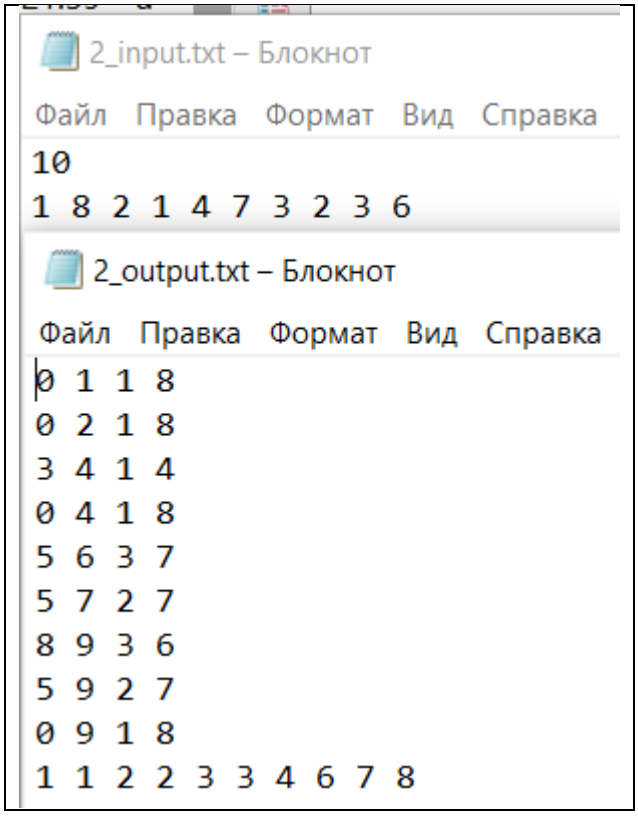
```
import time
import os, psutil


def merge(array, left_index, right_index, middle):
    left_copy = array[left_index:middle + 1]
    right_copy = array[middle+1:right_index+1]
    left_copy_index = 0
    right_copy_index = 0
    sorted_index = left_index

    while left_copy_index < len(left_copy) and right_copy_index <
len(right_copy):
```

```
        if left_copy[left_copy_index] <= right_copy[right_copy_index]:
            array[sorted_index] = left_copy[left_copy_index]
            left_copy_index = left_copy_index + 1
        else:
            array[sorted_index] = right_copy[right_copy_index]
            right_copy_index = right_copy_index + 1
        sorted_index = sorted_index + 1

    while left_copy_index < len(left_copy):
        array[sorted_index] = left_copy[left_copy_index]
        left_copy_index = left_copy_index + 1
        sorted_index = sorted_index + 1

    while right_copy_index < len(right_copy):
        array[sorted_index] = right_copy[right_copy_index]
        right_copy_index = right_copy_index + 1
        sorted_index = sorted_index + 1


def merge_sort(array, left_index, right_index):
    if left_index >= right_index:
        return
    middle = (left_index + right_index)//2
    merge_sort(array, left_index, middle)
    merge_sort(array, middle + 1, right_index)
    merge(array, left_index, right_index, middle)
    res = [left_index, right_index, array[left_index], array[right_index]]
    m.write(" ".join(map(str, res)))
    m.write("\n")


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("2_input.txt")
m = open("2_output.txt", "w")

num = int(f.readline())
string = f.readline()
numbers = list(map(int, string.split()))
merge_sort(numbers, 0, len(numbers) - 1)
m.write(" ".join(map(str, numbers)))

f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Мы написали функцию сортировки слиянием и для того, что разобраться поподробнее в сортировке выводит данные, которые просили в задаче.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример из задачи	0.002000699999999994 second	13.61328125 mb

Вывод по задаче:.Мы поподробнее разобрались в сортировке слиянием.

Задача №7. Поиск максимального подмассива за линейное время

Текст задачи:
Найти максимальный подмассив за линейное время, воспользовавшись следующими идеями, данными в условие

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \leq n \leq 10^5$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- Формат выходного файла (output.txt). В выходной файл надо вывести сумму максимального подмассива.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

Листинг кода

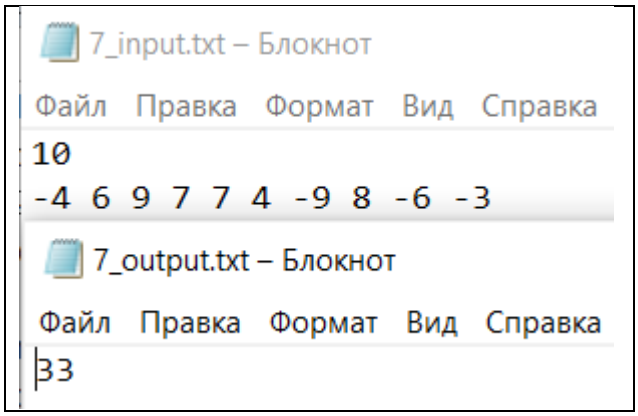
```
import time
import os, psutil
import random


def max_subarray_sum(my_array, array_size):
    max_till_now = my_array[0]
    max_ending = 0
    for n in range(0, array_size):
        max_ending = max_ending + my_array[n]
        if max_ending < 0:
            max_ending = 0
        elif max_till_now < max_ending:
            max_till_now = max_ending

    return max_till_now


def test(a, b):
    k = []
    for i in range(0, a):
        k.append(random.randint(-b, b))
    return k


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("7_input.txt", "w")
m = open("7_output.txt", "w")

f.write("10\n")
numbers = test(10, 10)
f.write(" ".join(map(str, numbers)))
m.write(str(max_subarray_sum(numbers, len(numbers))))

f.close()
m.close()
```

```
print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Я написала функцию поиска максимального подмассива, основываясь на идеях данных в условие.

Результат работы кода на примерах из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример из задачи	0.002000699999999994 second	13.62890625 mb

Вывод по задаче: Я узнала, как искать максимальный подмассив за линейное время.

Дополнительные задачи

Задача №4. Бинарный поиск

Текст задачи:

В этой задаче вы реализуете алгоритм бинарного поиска, который позволяет очень эффективно искать (даже в огромных) списках при условии, что список отсортирован. Цель - реализация алгоритма двоичного (бинарного) поиска.

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^5$) — число элементов в массиве, и последовательность $a_0 < a_1 < ... < a_{n-1}$ из n различных положительных целых чисел в порядке возрастания, $1 \le a_i \le 10^9$ для всех $0 \le i < n$. Следующая строка содержит число k, $1 \le k \le 10^5$ и k положительных целых чисел $b_0, ...b_{k-1}$, $1 \le b_j \le 10^9$ для всех $0 \le j < k$.
- Формат выходного файла (output.txt). Для всех i от 0 до $k - 1$ вывести индекс $0 \le j \le n - 1$, такой что $a_i = b_j$ или -1, если такого числа в массиве нет.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

Input.txt	5 1 5 8 12 13 5 8 1 23 1 11
Output.txt	2 0 -1 0 1

Листинг кода:

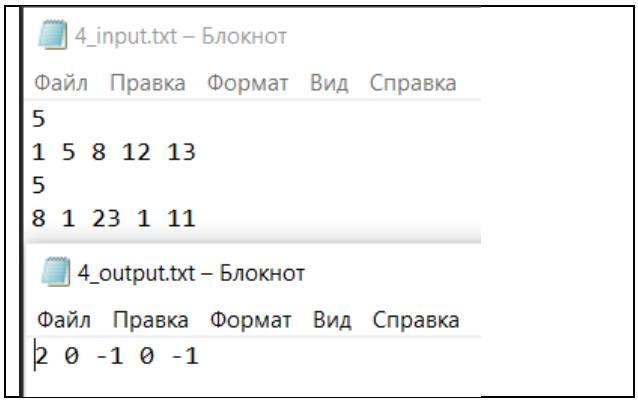
```
import time
import os, psutil


def binary_search(arr, val):
    first = 0
    last = len(arr)-1
    index = -1
    while (first <= last) and (index == -1):
        mid = (first+last)//2
        if arr[mid] == val:
            index = mid
        else:
            if val < arr[mid]:
                last = mid - 1
            else:
                first = mid + 1
    return index
```

```
t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("4_input.txt")
m = open("4_output.txt", "w")

num1 = int(f.readline())
string = f.readline()
array = list(map(int, string.split()))
num2 = int(f.readline())
string = f.readline()
find_indexes = list(map(int, string.split()))

indexes = []
for i in find_indexes:
    indexes.append(binary_search(array, i))

m.write(" ".join(map(str, indexes)))

f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Была написана функция бинарного поиска.

Результат работы кода на примере из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример из задачи	0.0022013999999999923 second	13.53125 mb

Вывод по задаче:.Мы научились писать функцию бинарного поиска и протестировали её.

Задача №4. Подсчёт инверсий

Текст задачи:
Дан массив целых чисел. Ваша задача — подсчитать число инверсий в нем. Подсказка: чтобы сделать это быстрее, можно воспользоваться модификацией
сортировки слиянием.

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^5$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- Формат выходного файла (output.txt). В выходной файл надо вывести число инверсий в массиве.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб

Input.txt	10 1 8 2 1 4 7 3 2 3 6
Output.txt	17

Листинг кода:

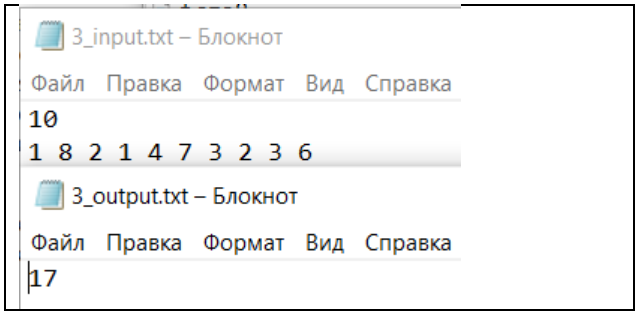
```
import time
import os, psutil


def merge_sort_help(arr, n):
    temp_arr = [0] * n
    return merge_sort(arr, temp_arr,
                      0, n - 1)


def merge_sort(arr, temp_arr, left, right):
    inv_count = 0
    if left < right:
        mid = (left + right) // 2
        inv_count += merge_sort(arr, temp_arr, left, mid)
        inv_count += merge_sort(arr, temp_arr, mid + 1, right)
        inv_count += merge(arr, temp_arr, left, mid, right)
    return inv_count


def merge(arr, temp_arr, left, mid, right):
    i = left
    j = mid + 1
    k = left
    inv_count = 0

    while i <= mid and j <= right:
        if arr[i] <= arr[j]:
            temp_arr[k] = arr[i]
            k += 1
```

```
            i += 1
        else:
            temp_arr[k] = arr[j]
            inv_count += (mid - i + 1)
            k += 1
            j += 1

    while i <= mid:
        temp_arr[k] = arr[i]
        k += 1
        i += 1

    while j <= right:
        temp_arr[k] = arr[j]
        k += 1
        j += 1

    for loop_var in range(left, right + 1):
        arr[loop_var] = temp_arr[loop_var]

    return inv_count


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("3_input.txt")
m = open("3_output.txt", "w")

num = int(f.readline())
string = f.readline()
numbers = list(map(int, string.split()))
m.write(str(merge_sort_help(numbers, num)))

f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Я использовала алгоритм сортировки слиянием при написание данной функции. Подсчет инверсий производился при слиянии массивов, основываясь на данном правиле: если элемент первой половины больше первого элемента второй половины, то он больше и всех элементов второй половины.

Результат работы кода на примере из текста задачи:



Тестирование алгоритма:

	Время выполнения	Затраты памяти
Пример из задачи	0.001357199999999989 second	13.59375 mb

Вывод по задаче:.Мы научились считать количество инверсий в массиве.

Вывод

Я узнала как работает сортировка слиянием и научилась применять ее на некоторых прикладных задачах, таких как бинарный поиск, поиск максимального подмассива и поиск числа инверсий в массиве.