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: Тема работы

Вариант 11

Выполнил:

Тимаков Егор

К3141

Проверила:

Артамонова В.Е.

Санкт-Петербург

2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Сортировка слиянием

1. 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 ($1 \leq n \leq 2 \cdot 10^4$) — 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.

2. Для проверки можно выбрать наихудший случай, когда сортируется массив размера 1000, 10^4 , 10^5 чисел порядка 10^9, отсортированных в обратном порядке; наилучший, когда массив уже отсортирован, и средний. Сравните, например, с сортировкой вставкой на этих же данных.

3. Перепишите процедуру Merge так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве. или перепишите процедуру Merge (и, соответственно, Merge-sort) так, чтобы в ней не использовались значения границ и середины - p, r и q.

Код:

```
import os
import psutil
from datetime import datetime

def merge(leftArray, rightArray):
    answer = []
    l = 0
```

```
    r = 0
    while l < len(leftArray) and r < len(rightArray):
        if leftArray[l] < rightArray[r]:
            answer.append(leftArray[l])
            l += 1
        else:
            answer.append(rightArray[r])
            r += 1
    if l < len(leftArray):
        answer += leftArray[l:]
    if r < len(rightArray):
        answer += rightArray[r:]
    return answer


def mergeSort(array):
    if len(array) == 1:
        return array
     middle = len(array) // 2  # делим массив на 2 для разбиения на части и суём в merge
                          return      merge(merge_sort(array[:middle]), merge_sort(array[middle:]))


process = psutil.Process(os.getpid())
starttime = datetime.now()

InputFile = open('input.txt')
length = InputFile.readline()
string = InputFile.readline().split()
InputFile.close()
array = []
for i in string:
    array.append(int(i))

OutputFile = open('output.txt', 'w')
OutputFile.write(str(merge_sort(array))[1:-1].replace(",", ""))

print(datetime.now() - starttime)
```

```
print(process.memory_info().rss / (1024 * 1024))
```

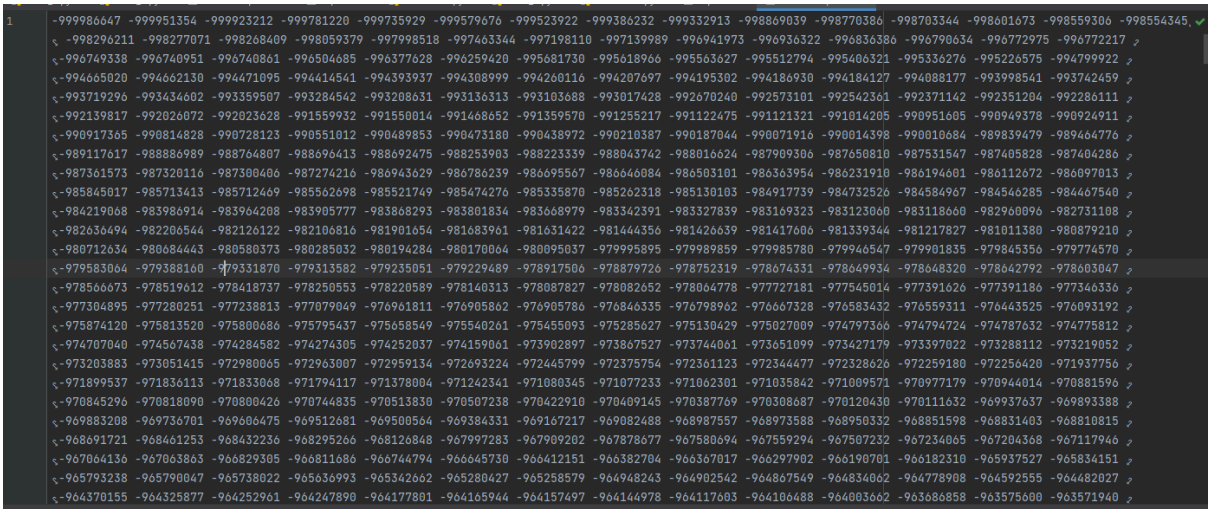
Объяснение:

Для реализации этого алгоритма я написал две функции: merge и mergeSort. в merge мы сравниваем элементы и меньший суем в итоговый массив. в mergeSort мы попросту делим массив и передаем его в функцию merge.

Результат работы кода на максимальных значениях:

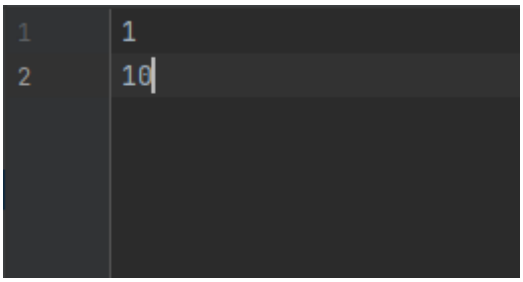
input.txt:

output.txt:



Результат работы кода при минимальных значениях:

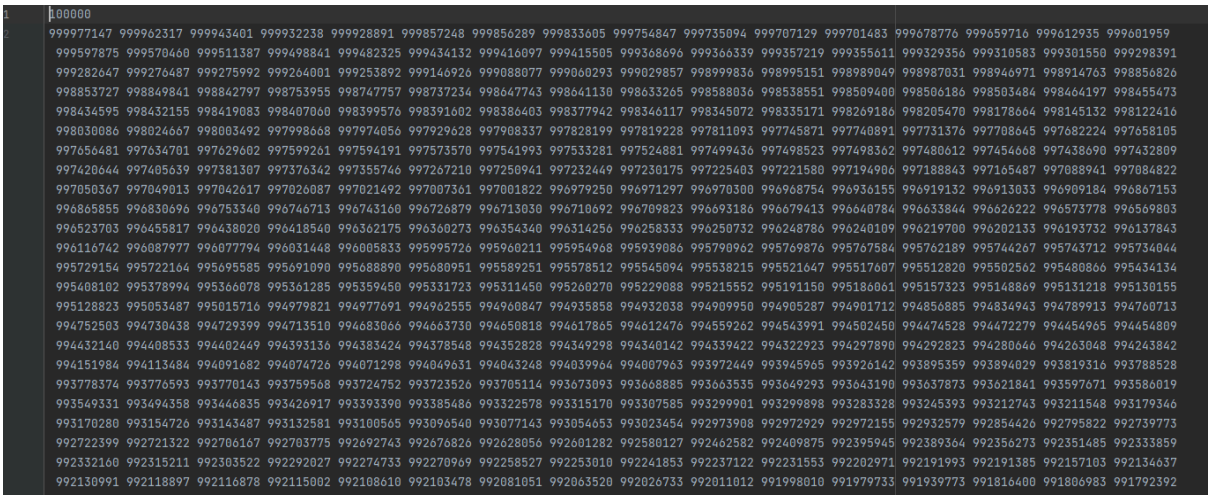
input.txt:



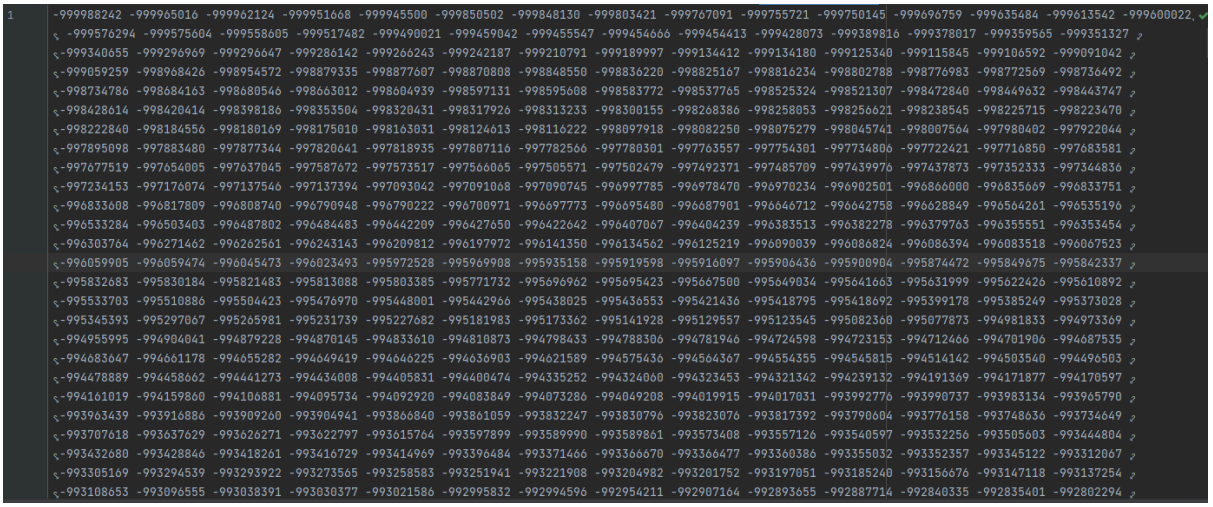
output.txt:

Результат работы кода при работе с массивом 10^5 отсортированный в обратном порядке:

input.txt:



output.txt:



Проверка задачи на время выполнения и затраты памяти

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	15
Верхняя граница диапазона значений входных данных из текста задачи	0.034	16
При 10^5 элементов отсортированных в обратном порядке.	0.260	26

Вывод по задаче:

При решении этой задачи пришлось изучить алгоритм сортировки слиянием

Задача №3. Число инверсий

Текст задачи:

Инверсией в последовательности чисел A называется такая ситуация, когда $i < j$, а $A_i > A_j$. Количество инверсий в последовательности в некотором роде определяет, насколько близка данная последовательность к отсортированной. Например, в сортированном массиве число инверсий равно 0, а в массиве, сортированном наоборот - каждые два элемента будут составлять инверсию (всего $n(n - 1)/2$). 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.

Код (Без рекурсии):

```
import os
import psutil
from datetime import datetime


def merge(leftArray, rightArray):
    answer = []
    global inv
    l = 0
    r = 0
    while l < len(leftArray) and r < len(rightArray):
        if leftArray[l] < rightArray[r]:
            answer.append(leftArray[l])
            l += 1
            inv += l

        else:
            answer.append(rightArray[r])
            r += 1
```

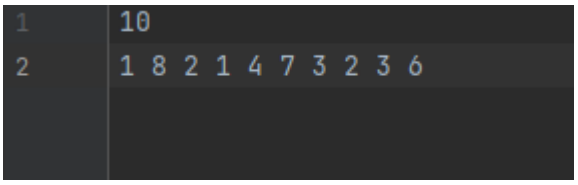
```
    if l < len(leftArray):
        answer += leftArray[l:]
    if r < len(rightArray):
        answer += rightArray[r:]
    return answer


def merge_sort(array):
    if len(array) == 1:
        return array
    middle = len(array) // 2
    return merge(merge_sort(array[:middle]), merge_sort(array[middle:]))


process = psutil.Process(os.getpid())
startTime = datetime.now()

InputFile = open('input.txt')
length = int(InputFile.readline())
string = InputFile.readline().split()
InputFile.close()
array = []
for i in string:
    array.append(int(i))

OutputFile = open('output.txt', 'w')
inv = 0
merge_sort(array)
OutputFile.write(str(inv))
OutputFile.close()

print(datetime.now() - startTime, process.memory_info().rss / (1024 * 1024))
```

Объяснение:

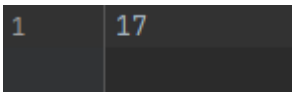
В данном алгоритме мы модифицировали алгоритм сортировки слиянием добавляя в функции Merge глобальную переменную inv, которая считает число инверсий.

Результат работы программы на примере из задачи:

input.txt:



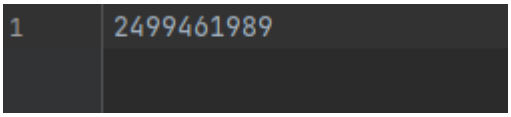
output.txt:



Результат работы кода на максимальных значениях:

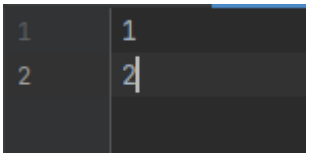
input.txt:

output.txt:



Результат работы кода на минимальных значениях:

input.txt



output.txt:

Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	14
Пример из задачи	0.001	14
Верхняя граница диапазона значений входных данных из текста задачи	0.43	26

Вывод по задаче:

В данной задаче мы реализовали подсчет инверсий с помощью сортировки слиянием.

Задача №4(Замена задачи №7). 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 $a0 < a1 < ... < an-1$ из n различных положительных целых чисел в порядке возрастания, $1 \le ai \le 10^9$ для всех $0 \le i < n$. Следующая строка содержит число k, $1 \le k \le 10^5$ и k положительных целых чисел b0, ...bk−1, $1 \le bj \le 10^9$ 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 ai = bj или -1, если такого числа в массиве нет.

• Ограничение по времени. 2сек.

• Ограничение по памяти. 256 мб.

Код:

```
import os
import psutil
from datetime import datetime


def Solution(array, value, start, stop):
    if start > stop:
        return -1
    else:
        mid = (start + stop) // 2
        if value == array[mid]:
            return mid
        elif value < array[mid]:
            return Solution(array, value, start, mid - 1)
        else:
            return Solution(array, value, mid + 1, stop)
```

```
process = psutil.Process(os.getpid())
startTime = datetime.now()

InputFile = open('input.txt')
lengthA = InputFile.readline()
arrayA = sorted([int(x) for x in InputFile.readline().split()])
lengthB = InputFile.readline()
arrayB = [int(x) for x in InputFile.readline().split()]


Outputfile = open('output.txt', 'w')
answer = ''
for i in range(len(arrayB)):
    x = Solution(arrayA, arrayB[i], 0, len(arrayA) - 1)
    arrayB[i] = x

for i in arrayB:
    answer += str(i) + ' '
Outputfile.write(f'{answer}')

print(datetime.now() - startTime, process.memory_info().rss / (1024 * 1024))
```

Объяснение:

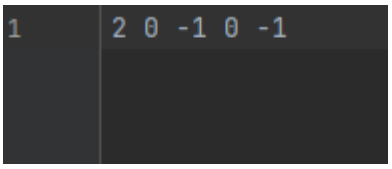
В данной программе мы также реализуем алгоритм бинарного поиска в функции Solution. В функции находим среднее значение, и если элемент меньше/больше среднего значения, то убираем большую/меньшую часть массива и рекурсивно вызываем функцию, пока среднее значение не будет равно элементу.

Результат работы программы на примере из задачи:

input.txt:

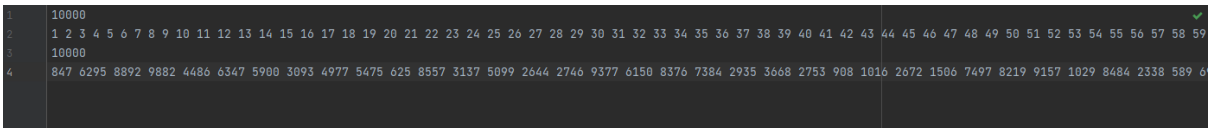
```
5
1 5 8 12 13
5
8 1 23 1 11
```

output.txt:

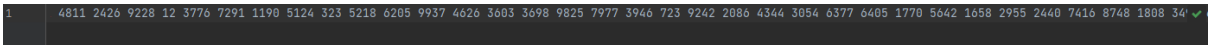


Результат работы кода на максимальных значениях:

input.txt:



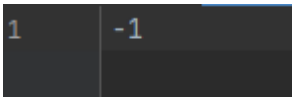
output.txt:



Результат работы кода на минимальных значениях:

input.txt:

output.txt:



Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	14
Пример из задачи	0.001	14
Верхняя граница диапазона значений входных данных из	0.037	15

текста задачи		

Вывод по задаче:

При выполнении данной задачи, потребовалось изучить алгоритм бинарного поиска

Вывод

В данной лабораторной я изучил алгоритм сортировкой слиянием и реализовал его. Также я изучил и реализовал алгоритм бинарного поиска.