МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: СОРТИРОВКА СЛИЯНИЕМ

Студент гр. 0304 _____ Докучаев Р.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург, 2021

Цель работы.

Изучить алгоритм сортировки слиянием и реализовать программу, которая выполняет эту сортировку

Задание.

На вход программе подаются квадратные матрицы чисел. Напишите программу, которая сортирует матрицы по возрастанию суммы чисел на главной диагонали с использованием алгоритма сортировки слиянием.

Формат входа.

Первая строка содержит натуральное число n - количество матриц. Далее на вход подаются n матриц, каждая из которых описана в формате: сначала отдельной строкой число m_i - размерность i-й по счету матрицы. После m строк по m чисел в каждой строке - значения элементов матрицы.

Формат выхода.

- Порядковые номера тех матриц, которые участвуют в слиянии на очередной итерации алгоритма. Вывод с новой строки для каждой итерации.
- Массив, в котором содержатся порядковые номера матриц, отсортированных по возрастанию суммы элементов на диагонали. Порядковый номер матрицы - это её номер по счету, в котором она была подана на вход программе, нумерация начинается с нуля.

Основные теоретические положения.

Были использованы библиотеки *math* и *copy* языка Python. Также был использован алгоритм сортировки слиянием, который заключается в разделении исходного массива на более мелкие массивы, затем происходит сравнение мелких единиц, а после этого массив снова объединяется, но он уже в отсортированном виде. Также была использована утилита *pytest* для тестирования работы программы.

Выполнение работы.

1. Были подключены библиотеки *copy* и *math* в файле main.py
2. Была создана функцию-обёртку для вызова функции *mergesort* не нулевая, то массива заполняется значениями None.

3. Была реализована рекурсивная функция mergesort, которая разделяет массив таким образом, чтобы левая часть всегда была меньше правой, затем при помощи функции merge происходит сравнение элементов разбитых массивов, после чего отсортированные части склеиваются и выводятся на экран пользователя.
4. Была реализована функция merge, которая выбирает минимальный элемент двух массивов для проведения сортировки элементов и дальнейшего слияния.
5. Был создан тестовый модель test.py, который на примерах проверяет корректность выполнения сортировки.

Исходный код программы представлен в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	3 2 1 2 1 31 3 1 1 1 1 11 1 1 1 -1 5 1 2 0 1 -1 1 2 0 1 -1 1 2 0 1 -1		OK

	1 2 0 1 -1 1 2 0 1 -1		
2.	3 2 -62 -8 -1 97 3 -98 -84 28 32 -85 -33 96 -68 -99 2 15 81 67 68	1 2 1 0 2 1 0 2	OK

Выводы.

Была изучена сортировка слиянием, а также была реализована программа, которая выполняет сортировку слиянием для заданного массива.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import math
import copy

def _mergesort(arr):
    if len(arr) > 0:
        temp = [None]*len(arr)
        mergesort(arr, temp, 0, len(arr)-1)

def mergesort(arr, temp, start, end):
    if start >= end:
        return
    middle = math.ceil((start+end)/2)
    mergesort(arr, temp, start, middle-1)
    mergesort(arr, temp, middle, end)

    temp = copy.deepcopy(arr)
    merge(arr, temp, start, end)
    for i in range(start, end+1):
        print(arr[i][1], end=' ')
    print('')

def merge(out_, in_, start, end):
    middle = math.ceil((start + end)/2)
    i = start
    j = middle
    k = start
    while i < middle and j <= end:
        if in_[i][0] > in_[j][0]:
            out_[k] = in_[j]
            j += 1
        elif in_[i][0] < in_[j][0]:
            out_[k] = in_[i]
            i += 1
        elif in_[i][1] > in_[j][1]:
            out_[k] = in_[j]
            j += 1
        else :
            out_[k] = in_[i]
            i += 1
        k += 1
    while i < middle:
        out_[k] = in_[i]
        i += 1
        k += 1
    while j < end:
        out_[k] = in_[j]
        j += 1
        k += 1

if __name__ == '__main__':
    n = int(input())
    list_ = [None]*n
```

```
for i in range (0,n):
    m = int(input())
    t = 0
    for j in range (0,m):
        line = list(map(int, input().split()))
        t += line[j]
    list_[i] = (t, i)
_mergesort(list_)
for i in range (0, n):
    print(list_[i][1], end = ' ')
print('')
```