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

Студентка гр. 1383 _____ Седова Э.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2022

Цель работы.

Ознакомиться с алгоритмами сортировки.

Задание.

Сортировка слиянием.

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.

Выполнение работы.

Программа принимает входные данные от пользователя. С помощью цикла *for* высчитывается сумма чисел диагонали каждой матрицы. Далее, матрицы сортируются по возрастанию суммы чисел на главной диагонали с помощью функции *merge(arr, dict)*. Функция *merge(arr, dict)* принимает на вход список порядковых номеров матриц и словарь. Ключи словаря – порядковые номера матриц, значения – их длины. В данной функции происходит сортировка слиянием рекурсивным способом. Сначала функция вызывает сама себя для

левой половины стека, затем для правой половины. Если *merge()* передаёт список из одного или нуля элементов, то она вернёт его обратно, так как он уже является отсортированным. При завершении работы будет возвращён упорядоченный по возрастанию список, а в списке *results* будут храниться все промежуточные списки длины больше единицы

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	[0, 1, 2], {0: 100, 1: 200, 2: 300}	[0, 1, 2]
2.	[0, 1, 2], {0: 35, 1: -200, 2: 1}	[1, 2, 0]
3.	[0, 1, 2, 3], {0: 3, 1: -200, 2: 1, 3: -9}	[1, 3, 2, 0]
4.	[0, 1, 2], {0: 1, 1: -35, 2: -35}	[1, 2, 0]

Выводы.

Были получены навыки реализации алгоритмов сортировки на практике.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
def merge(arr, dict):
    if len(arr) == 1:
        return arr
    middle = len(arr) // 2
    left, right = arr[:middle], arr[middle:]
    merge(left, dict)
    merge(right, dict)
    index_left = index_right = index = 0
    result = [0] * (len(left) + len(right))
    while index_left < len(left) and index_right < len(right):
        if dict[left[index_left]] <= dict[right[index_right]]:
            result[index] = left[index_left]
            index_left += 1
        else:
            result[index] = right[index_right]
            index_right += 1
        index += 1
    while index_left < len(left):
        result[index] = left[index_left]
        index_left += 1
        index += 1
    while index_right < len(right):
        result[index] = right[index_right]
        index_right += 1
        index += 1
    for i in range(len(arr)):
        arr[i] = result[i]
        print(arr[i], end=' ')
    print()
    return arr


if __name__ == "__main__":
    n = int(input())
    arr = []
    dict={}
    for i in range(n):
        m = int(input())
        summa = 0
        numbers = 0
        for j in range(m):
            numbers = list(map(int, input().split()))
            summa += numbers[j]
        arr.append(i)
        dict[i]=summa
    for i in range(len(merge(arr,dict))):
        print(arr[i], end=' ')
    print()
```