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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
ТЕМА: «СОРТИРОВКИ».

Студент гр. 1304 _____ Байков Е.С.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2022

Цель работы

Изучить работу алгоритма сортировки слиянием.

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.

Выполнение работы

Был создан специальный класс *Matrix* хранящий в себе диагональную сумму матрицы, ее размер и номер по счету, написан метод *set_diag()*, который устанавливает диагональную сумму матрицы также переопределенные операторы сравнения для данного типа и метод преобразования в строку.

На вход программе с помощью функции *input()* и функции *int()* строка преобразуется в число и переменной *n* присваивается это значение – количество строк. Затем создается список *matrix_list*, в котором будут хранится экземпляры класса *Matrix*. Затем с помощью двух циклов *for* считывается размер матрицы,

она добавляется в список, а после высчитывается ее диагональная сумма и устанавливается для данной матрицы. Затем вызывается функция *merge()* с переданным в него списком *matrix_list*. Происходит сортировка слиянием по суммам на главной диагонали списка матриц. Затем с помощью функции *join()*, в которую передается преобразованный список, хранящий в себе не матрицы а их индексы, и функции *print()* происходит вывод на экран результата.

Функции:

merge() – функция которая принимает на вход массив, который необходимо отсортировать. Сначала идет проверка на то что массив является массивом длиной 1. С помощью переменной *middle* определяется середина списка, а затем берутся два среза исходного списка – *left* и *right*. Определяются индексы *index_left, index_right,* index равные 0. Переменную *result* объявляется как список из 0 длиной равной сумме длин левой и правой частей.

С помощью цикла *while* происходит обходит по левой и правой частям, пока одна из них не закончится. Осуществляется сравнение двух элементов *left[index_left] <= right[index_right]*. При положительном результате в *result* записывается элемент из левого списка, а *index_left* увеличивается на 1. Иначе то же самое будет происходить с правой частью. Затем значение index увеличивается на 1.

С помощью последующих циклов *while* рассматривается случай если одно из частей не закончилась. В этом случае оставшиеся элементы из той или иной части переписываются в *result*.

В конце с помощью двух циклов *for* каждый элемент из *arr* становиться равным *result[i]*, и выводится какие именно элементы были поменяны местами. А затем возвращается *arr*.

create_matrix_list_from_str – вспомогательная функция преобразующая строку в список матриц. Сначала строка делится на под строки по символу переноса строки, создается список матриц. Первый элемент списка строк – количество матриц. С помощью среза из списка *arr* исключается нулевой

элемент. С помощью двух циклов *for* создается список матриц, а из *arr* исключаются строки. Возвращается список матриц.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	4 2 1 4 4 12 4 11 2 3 4 2 -12 2 3 143 -11 2 3 3 4 5 40 3 1 2 3 3 2 1 2 1 3 2 2 2 4 24	0 1 2 3 2 0 3 1 2 0 3 1	Проверка при четном количестве матриц
2.	3 2 2 4 4 -12 3 3 4 3 4 -25 12 0 0 1 3 12 3 4 4 44 12 21 2 -123	2 1 2 1 0 2 1 0	Проверка при отрицательных суммах
3.	3 2 1 2 1 31 3 1 1 1 1 11 1 1 1 -1 5	2 1 2 1 0 2 1 0	Проверка при нечетном количестве матриц

	1 2 0 1 -1 1 2 0 1 -1 1 2 0 1 -1 1 2 0 1 -1 1 2 0 1 -1		

Выводы

Был изучен алгоритм сортировки слиянием. Была разработана программа сортирующая список матриц по сумме на главной диагонали.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
class Matrix:
    def __init__(self, size, number):
        self.size = size
        self.diag = 0
        self.num = number

    def set_diag(self, diag):
        self.diag = diag

    def __eq__(self, other) -> bool:
        return self.diag == other.diag

    def __lt__(self, other):
        return self.diag < other.diag
    def __le__(self, other):
        return self.diag <= other.diag
    def __ne__(self, other):
        return self.diag != other.diag
    def __gt__(self, other):
        return self.diag > other.diag
    def __ge__(self, other):
        return self.diag >= other.diag

    def __str__(self) -> str:
        return f'{self.num}'

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

def create_matrix_list_from_str(string:str):
    arr = string.split('\n')
    matrix_list = []
    n = int(arr[0])
    arr = arr[1:]
    for i in range(n):
        size = int(arr[0])
        matrix_list.append(Matrix(size, i))
        diag = 0
        matrix = []
        for j in range(size):
            elem = list(map(int, arr[j+1].split()))
            matrix.append(elem)
            diag += matrix[j][j]
        matrix_list[i].set_diag(diag)
        arr = arr[size + 1:]
    return matrix_list


if __name__ == '__main__':
    n = int(input())
    matrix_list = []

    for i in range(n):
        size = int(input())
        matrix_list.append(Matrix(size, i))
        diag = 0
        matrix = []
        for j in range(size):
            elem = list(map(int, input().split()))
            matrix.append(elem)
            diag += matrix[j][j]
        matrix_list[i].set_diag(diag)

    merge(matrix_list)

    print(' '.join(list(map(str, matrix_list))))
```

Название файла: tests.py

```
from main import Matrix, merge, create_matrix_list_from_str

def test():
    string = '3\n2\n1 2\n1 31\n3\n1 1\n1 11 1\n1 1 -1\n5\n1 2 0
1 -1\n1 2 0 1 -1\n1 2 0 1 -1\n1 2 0 1 -1\n1 2 0 1 -1'
    matrix_list = create_matrix_list_from_str(string)
    merge(matrix_list)
    expected_result = [Matrix(5, 2), Matrix(3, 1), Matrix(2, 0)]
    expected_result[0].set_diag(3)
    expected_result[1].set_diag(11)
    expected_result[2].set_diag(32)
    assert matrix_list == expected_result
```