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: Сортировки

Студент гр. 0304 _____ Мажуга Д. Р.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2021

Цель работы.

Изучить метод 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 mi - 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.

Сортировка слиянием — один из самых известных алгоритмов сортировки. Алгоритм делит исходный массив на две равные части, если массив нечетной длины, то одна из частей больше другой на еденицу, полумассивы делятся снова, так повторяется, пока каждый массив не будет состоять только из одного элемента, затем одинарные массивы объединяются попарно по возрастанию, пары по возрастанию объединяются в массивы из четырех элементов и т.д., пока не будет получен отсортированный массив исходных элементов.

Выполнение работы.

В данной работе был реализован класс Matrix для сравнения сум главной диагонали. Был реализован конструктор в котором мы непосредственно храним нашу матрицу, а такжже ее размерность и порядковый номер.

В данной работе были реализованы следующие методы:

1. Метод *print_matrix()* - в данном методе мы выводим нашу матрицу в

терминал , мы в двойном цикле выводим элементы матрицы каждой строки, данный метод был реализован следующим образом: в двойном цикле выводим элементы матрицы каждой строки.

2. Метод *get_number()* - это метод возвращающий порядковый номер матрицы.
3. Метод *get_trace_sum()* - в данном методе было реализованно получение суммы главной диагонали матрицы, путем сложения элементов лежащих на главной диагонали в цикле.
4. Методы перегрузки операторов стравнений и равенства, были сделаны для того, чтобы мы могли корректно сравнивать матрицы.

Также, были реализованна функции *merge_sort(arr)* и *merge(left, right)* - в данных функция мы непосредственно сортируем наши матрицы слиянием.

Функция *merge_sort()* - в данной функции мы передаём массив с нашими матрицами, затем мы проверяем длину массива если же длина равна нулю или 1, то мы возвращаем наш массив, поскольку сортировать нечего. В том случае если длина массива больше или равна двух, мы разделяем массив на две части(левую и правую) далее для каждой из частей мы опять вызваем функцию *merge_sort()*, после чего отсортированные части мы предаём в функцию *merge()*.

Функия *merge()* - в данную функцию мы передайм наши левую и правую части, затем мы создаём массив result в который в последствии будет записывать результат слияния. Далее мы в цикле while сравниваем элементы левых и правых частей, на каждом этапе сравнения мы смотрим, больше ли *i*-тый элемент левой части, *j*-того элемента правой, после сравнения мы выводим промежуточный результат в терминал, записываем результат сравнения в result и увеличиваем *i* или *j*, в зависимости от того какая часть оказалась больше. Поскольку в таком сравнения мы можем не пройти одной из частей до конца, то все элементы не пройденной части добавляются в *result.*

Тестирование.

Были написанны тесты, необходимые для проверки корректности программы, сортирующей массив матриц методом слияния. Результаты тестирования представлены в табл. 1.

№	Входные данные	Выходные данные	Комментарий
1	3 2 -62 -8 -1 97 3 -98 -84 28 32 -85 -33 96 -68 -99 2 15 81 67 68	1 0 2	OK
2	1 3 -92 77 -12 73 81 100 -11 44 -55	0	OK
3	3 2 1 0 0 1 2 1 77 -12 1 2 1 -4 22 1	0 1 2	OK

Таблица 1 – Результаты тестирования

Вывод.

Результатом проведённой работы является программа, принимающая на вход количество матриц, размерность каждой из них, сами матрицы, после чего сортирующая их по возрастанию следа матрицы методом слияния.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: merge_sort.py

```
from Matrix import Matrix

def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if(left[i] <= right[j]):
            result.append(left[i])
            print(left[i].get_number(), end=' ')
            i += 1
        else:
            result.append(right[j])
            print(right[j].get_number(), end=' ')
            j += 1
    while i < len(left):
        result.append(left[i])
        print(left[i].get_number(), end=' ')
        i += 1
    while j < len(right):
        result.append(right[j])
        print(right[j].get_number(), end=' ')
        j += 1
    print()
    return result

def merge_sort(arr):
    if len(arr) < 2:
        return arr[:]
    else:
        middle = int(len(arr) / 2)
        left = merge_sort(arr[:middle])
        right = merge_sort(arr[middle:])
        return merge(left, right)

if __name__ == '__main__':
    number_matr = int(input())
    dimension_matr = int(input())
    matrixes = [Matrix()] * number_matr

    for i in range(number_matr):
        matr = []
        for j in range(dimension_matr):
            matr.append(list(map(int, input().split())))
        matrixes[i] = Matrix(matr, dimension_matr, i)
        if i + 1 < number_matr:
            dimension_matr = int(input())

    sort_value = merge_sort(matrixes)
```

```
    for elem in sort_value:
        print(elem.get_number(), end=' ')
```

Название файла: Matrix.py

```
class Matrix:
    def __init__(self, lst_matr=None, dimension=None, number=None):
        self.matr = lst_matr
        self.dimension = dimension
        self.number = number

    def __len__(self):
        return self.dimension

    def __eq__(self, other):
        return self.get_trace_sum() == other.get_trace_sum()

    def __gt__(self, other):
        return self.get_trace_sum() > other.get_trace_sum()

    def __ge__(self, other):
        return self.get_trace_sum() >= other.get_trace_sum()

    def __lt__(self, other):
        return self.get_trace_sum() < other.get_trace_sum()

    def __le__(self, other):
        return self.get_trace_sum() <= other.get_trace_sum()

    def print_matrix(self):
        for i in range(self.dimension):
            for j in range(self.dimension):
                print(self.matr[i][j], end=' ')
            print()
        print(self.get_trace_sum())

    def get_number(self):
        return self.number

    def get_trace_sum(self):
        sum = 0
        for i in range(self.dimension):
            sum += self.matr[i][i]
        return sum
```

Название файла: tests.py

```
from merge_sort import Matrix, merge_sort

class TestCauses:

    @staticmethod
    def frst_test():
        matr1 = [[-62, -8], [-1, 97]]
```

```
        matr2 = [[-98, -84, 28], [32, -85, -33], [96, -68, -99]]
        matr3 = [[15, 81], [67, 68]]
        check = []
        gues_value = [1, 0, 2]

        matixes = [Matrix(matr1, 2, 0), Matrix(matr2, 3, 1), Matrix(matr3, 2, 2)]
        sort = merge_sort(matixes)
        for item in sort:
            check.append(item.get_number())
        assert check == gues_value
        print('first test: OK')

    @staticmethod
    def scnd_test():
        matr = [[-92, 77, -12], [73, 81, 100], [-11, 44, -55]]
        check = []
        gues_value = [0]

        matrix = [Matrix(matr, 3, 0)]
        sort = merge_sort(matrix)
        for item in sort:
            check.append(item.get_number())
        assert check == gues_value
        print('second test: OK')

    @staticmethod
    def thrd_test():
        matr1 = [[1, 0], [0, 1]]
        matr2 = [[1, 77], [-12, 1]]
        matr3 = [[1, -4], [22, 1]]

        check = []
        gues_value = [0, 1, 2]

        matrixes = [Matrix(matr1, 2, 0), Matrix(matr2, 2, 1), Matrix(matr3, 2, 2)]
        sort = merge_sort(matrixes)
        for item in sort:
            check.append(item.get_number())
        assert check == gues_value
        print('third test: OK')

if __name__ == '__main__':
    case = TestCauses
    case.frst_test()
    case.scnd_test()
    case.thrd_test()
```