Министерство цифрового развития, связи и массовых коммуникаций
Государственное образовательного учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Задачи для самостоятельного решения

по дисциплине «Структура и алгоритмы обработки данных»

Выполнил студент группы БФИ 1901:

Гребешечников Иван

Проверил:

Кутейников Иван Алексеевич

Москва 2021

Задание

Задание №2:

Написать генератор случайных матриц(многомерных), который принимает опциональные параметры **m, n, min_limit, max_limit,** где **m** и **n** указывают размер матрицы, а **min_lim** и **max_lim** - минимальное и максимальное значение для генерируемого числа . По умолчанию при отсутствии параметров принимать следующие значения:

m = 50

n = 50

min_limit = -250

max_limit = 1000 + (номер своего варианта)

Задание №3:

Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

Методы:

Выбором	Вставкой	Обменом	Шелла	Турнирная	Быстрая сортировка	Пирамидальная

Код программы

```
package com.company;
import java.io.IOException;
import java.util.Scanner;

public class Lab1 {
    public static void selectionSort(int[] arr){

        for (int i = 0; i < arr.length; i++) {
        /*Предполагаем, что первый элемент (в каждом
           подмножестве элементов) является минимальным */
            int min = arr[i];
            int min_i = i;
        /*В оставшейся части подмножества ищем элемент,
           который меньше предположенного минимума*/
            for (int j = i+1; j < arr.length; j++) {
                //Если находим, запоминаем его индекс
                if (arr[j] < min) {
```

```
                    min = arr[j];
                    min_i = j;
                }
            }
        /*Если нашелся элемент, меньший, чем на текущей позиции,
          меняем их местами*/
            if (i != min_i) {
                int tmp = arr[i];
                arr[i] = arr[min_i];
                arr[min_i] = tmp;
            }
        }
    }
    public static void insertionSort(int[] array) {
        for (int i = 1; i < array.length; i++) {
            int current = array[i];
            int j = i - 1;
            while(j >= 0 && current < array[j]) {
                array[j+1] = array[j];              // передвигаем больший
элемент вправо
                j--;
            }
            // в этой точке мы вышли, так что j так же -1
            // или в первом элементе, где текущий >= a[j]
            array[j+1] = current;
        }
    }
    public static void SwapSort(int[] arr){

        for(int i = arr.length-1 ; i > 0 ; i--){
            for(int j = 0 ; j < i ; j++){

            if( arr[j] > arr[j+1] ){
                int tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}
    public static void ShellSort(int [] a){
            int temp;
            int h = 0;//величина интервала

            //вычисляем исходное значение интервала
            while(h <= a.length/3)
                h = 3*h + 1;

            for(int k = h; k > 0; k = (k-1)/3)
                for(int i = k; i < a.length; i++)
                {
                    temp = a[i];
                    int j;
                    for(j = i; j >= k; j -= k)
                    {
                        if(temp < a[j - k])
                            a[j] = a[j - k];
                        else
                            break;
                    }
                    a[j] = temp;
                }
        }
    public static void quickSort(int[] a, int leftBorder, int rightBorder) {
```

```
        int leftMarker = leftBorder;
        int rightMarker = rightBorder;
        int pivot = a[(leftMarker + rightMarker) / 2];
        do {
            // Двигаем левый маркер слева направо пока элемент меньше, чем
pivot
            while (a[leftMarker] < pivot) {
                leftMarker++;
            }
            // Двигаем правый маркер, пока элемент больше, чем pivot
            while (a[rightMarker] > pivot) {
                rightMarker--;
            }
            // Проверим, не нужно обменять местами элементы, на которые
указывают маркеры
            if (leftMarker <= rightMarker) {
                // Левый маркер будет меньше правого только если мы должны
выполнить swap
                if (leftMarker < rightMarker) {
                    int tmp = a[leftMarker];
                    a[leftMarker] = a[rightMarker];
                    a[rightMarker] = tmp;
                }
                // Сдвигаем маркеры, чтобы получить новые границы
                leftMarker++;
                rightMarker--;
            }
        } while (leftMarker <= rightMarker);

        // Выполняем рекурсивно для частей
        if (leftMarker < rightBorder) {
            quickSort(a, leftMarker, rightBorder);
        }
        if (leftBorder < rightMarker) {
            quickSort(a, leftBorder, rightMarker);
        }
    }
    // Процедура для преобразования в двоичную кучу поддерева с корневым
узлом i, что является
// индексом в arr[]. n - размер кучи
    static void heapify(int arr[], int n, int i)
    {
        int largest = i; // Инициализируем наибольший элемент как корень
        int l = 2*i + 1; // левый = 2*i + 1
        int r = 2*i + 2; // правый = 2*i + 2

        // Если левый дочерний элемент больше корня
        if (l < n && arr[l] > arr[largest])
            largest = l;

        // Если правый дочерний элемент больше, чем самый большой элемент на
данный момент
        if (r < n && arr[r] > arr[largest])
            largest = r;
        // Если самый большой элемент не корень
        if (largest != i)
        {
            int swap = arr[i];
            arr[i] = arr[largest];
            arr[largest] = swap;

            // Рекурсивно преобразуем в двоичную кучу затронутое поддерево
            heapify(arr, n, largest);
        }
```

```
    }
    public static void piramSort(int arr[]){
        int n = arr.length;

        // Построение кучи (перегруппируем массив)
        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(arr, n, i);

        // Один за другим извлекаем элементы из кучи
        for (int i=n-1; i>=0; i--)
        {
            // Перемещаем текущий корень в конец
            int temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;

            // Вызываем процедуру heapify на уменьшенной куче
            heapify(arr, i, 0);
        }
    }


    public static void main(String[] args) throws IOException,
InterruptedException {
        //System.out.println("Hello world");

        Scanner scanner = new Scanner(System.in);
        if (scanner.hasNextInt()) {

            int n = scanner.nextInt();
            int m = scanner.nextInt();

        int a [][] = new int [n][m];
        for (int i = 0;i<n;i++) {
            for (int j = 0; j < m; j++) {
                a[i][j] = -250 + (int) (Math.random() *1007);
            }
        }
            System.out.println("Изначальная матрица");
            for (int i=0; i<n; i++) {
                for (int j = 0; j < m; j++) {
                    System.out.print(a[i][j] + " ");
                }
                System.out.print("\n");
            }
   System.out.println("Выбором");
            long start = System.nanoTime();
        for (int s =0; s<n;s++) {
            int aa[] = new int[m];

            for (int st = 0; st < m; st++) {
                aa[st] = a[s][st];
            }

           selectionSort(aa);
            for (int l = 0; l<m;l++){
                System.out.print(aa[l]+" ");
            }
            System.out.println("\n");
        }
            Thread.sleep(1000);
            long finish = System.nanoTime();
            long elapsed = finish - start;
            System.out.println("Прошло времени, нс: " + elapsed);
```

```
System.out.println("Вставкой");

        long start1 = System.nanoTime();
        for (int s =0; s<n;s++) {
            int aa[] = new int[m];

            for (int st = 0; st < m; st++) {
                aa[st] = a[s][st];
            }

            insertionSort(aa);
            for (int l = 0; l<m;l++){
                System.out.print(aa[l]+" ");
            }
            System.out.println("\n");
        }
        Thread.sleep(1000);
        long finish1 = System.nanoTime();
        long elapsed1 = finish1 - start1;
        System.out.println("Прошло времени, нс: " + elapsed1);

  System.out.println("Обменом");

        long start2 = System.nanoTime();
        for (int s =0; s<n;s++) {
            int aa[] = new int[m];

            for (int st = 0; st < m; st++) {
                aa[st] = a[s][st];
            }

            SwapSort(aa);
            for (int l = 0; l<m;l++){
                System.out.print(aa[l]+" ");
            }
            System.out.println("\n");
        }
        Thread.sleep(1000);
        long finish2 = System.nanoTime();
        long elapsed2 = finish2 - start2;
        System.out.println("Прошло времени, нс: " + elapsed2);

  System.out.println("Шелла");

        long start3 = System.nanoTime();
        for (int s =0; s<n;s++) {
            int aa[] = new int[m];

            for (int st = 0; st < m; st++) {
                aa[st] = a[s][st];
            }

            ShellSort(aa);
            for (int l = 0; l<m;l++){
                System.out.print(aa[l]+" ");
            }
            System.out.println("\n");
        }
        Thread.sleep(1000);
        long finish3 = System.nanoTime();
        long elapsed3 = finish3 - start3;
        System.out.println("Прошло времени, нс: " + elapsed3);
```

```
     System.out.println("Быстрая");

            long start4 = System.nanoTime();
            for (int s =0; s<n;s++) {
                int aa[] = new int[m];

                for (int st = 0; st < m; st++) {
                    aa[st] = a[s][st];
                }

                quickSort(aa,0,aa.length-1);
                for (int l = 0; l<m;l++){
                    System.out.print(aa[l]+" ");
                }
                System.out.println("\n");
            }
            Thread.sleep(1000);
            long finish4 = System.nanoTime();
            long elapsed4 = finish4 - start4;
            System.out.println("Прошло времени, нс: " + elapsed4);

     System.out.println("Пирамидальная");
            long start5 = System.nanoTime();
            for (int s =0; s<n;s++) {
                int aa[] = new int[m];

                for (int st = 0; st < m; st++) {
                    aa[st] = a[s][st];
                }

                piramSort(aa);
                for (int l = 0; l<m;l++){
                    System.out.print(aa[l]+" ");
                }
                System.out.println("\n");
            }
            Thread.sleep(1000);
            long finish5 = System.nanoTime();
            long elapsed5 = finish5 - start5;
            System.out.println("Прошло времени, нс: " + elapsed5);
        }
    }
}
```

Вывод: в ходе выполнения данной работы я узнал о методах сортировки, их плюсах и минусах, сложностях алгоритмов, написал каждый из этих алгоритмов на языке программирования.