**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

**Отчет по лабораторной работе №1**

по дисциплине «Структуры и алгоритмы обработки данных»

на тему «Методы сортировки»

Выполнил: студент группы БВТ1901

Перевозчиков С. В.

Руководитель:

Мелехин А. А.

Москва 2021

Цель работы: изучить основные виды сортировок и написать их реализацию на одном из языков программирования.

Техническое задание:

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)

Реализовать  методы  сортировки  строк  числовой  матрицы  в  соответстви с заданием.

Оценить  время  работы  каждого  алгоритма  сортировки  и  сравнить  его  современем  стандартной функции сортировки.

Испытания проводить на сгенерированных матрицах.

Методы сортировки:

1. Выбором
2. Вставкой
3. Обменом
4. Шелла
5. Турнирная
6. Быстрая сортировка
7. Пирамидальная

Выполнение задания:

Генератор матриц:

int[][] matrix;

int m, n, min\_limit, max\_limit;

try

{

if(args.length >= 5)

{

m = Integer.parseInt(args[1]);

n = Integer.parseInt(args[2]);

min\_limit = Integer.parseInt(args[3]);

max\_limit = Integer.parseInt(args[4]);

matrix = new int[m][n];

}

else

{

m = 50;

n = 50;

min\_limit = -250;

max\_limit = 1017;

matrix = new int[n][m];

}

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

{

matrix[i][j] = (int)(Math.random()\*(max\_limit - min\_limit + 1) + min\_limit);

}

}

System.out.println(Arrays.deepToString(matrix));

1. Сортировка выбором.

Код:

public static int[][] selection(int[][] matrix)

{

for(int i = 0; i < matrix.length; i++)

{

//int min = matrix[i][0];

//System.out.println(matrix[i].length);

for(int j = 0; j < matrix[i].length; j++)

{

int min = matrix[i][j], index = j, changer;

for(int k = j; k < matrix[i].length; k++)

{

if(matrix[i][k] < min)

{

min = matrix[i][k];

index = k;

}

}

changer = matrix[i][j];

matrix[i][j] = min;

matrix[i][index] = changer;

}

}

System.out.println(Arrays.deepToString(matrix));

return matrix;

}

1. Сортировка вставкой.

Код:

public static int[][] insertion(int[][] matrix)

{

for(int i = 0; i < matrix.length; i++)

{

for(int j = 1; j < matrix[i].length; j++)

{

int num = matrix[i][j], changer;

for(int k = j-1; k >= 0; k--)

{

if(matrix[i][k] > num)

{

changer = matrix[i][k];

matrix[i][k] = num;

matrix[i][k+1] = changer;

}

}

}

}

System.out.println(Arrays.deepToString(matrix));

return matrix;

}

1. Сортировка обменом.

Код:

public static int[][] bubble(int[][] matrix)

{

for(int i = 0; i < matrix.length; i++)

{

for(int j = matrix[i].length; j > 0; j--)

{

int changer;

for(int k = 0; k < j-1; k++)

{

if(matrix[i][k] > matrix[i][k+1])

{

changer = matrix[i][k+1];

matrix[i][k+1] = matrix[i][k];

matrix[i][k] = changer;

}

}

}

}

System.out.println(Arrays.deepToString(matrix));

return matrix;

}

1. Сортировка Шелла.

public static int[][] shell(int[][] matrix)

{

for(int i = 0; i < matrix.length; i++)

{

for(int d = matrix[i].length / 2; d > 1; d = d/2)

{

int changer;

for(int j = 0; j < matrix[i].length - d; j++)

{

if(matrix[i][j] > matrix[i][j+d])

{

changer = matrix[i][j+d];

matrix[i][j+d] = matrix[i][j];

matrix[i][j] = changer;

}

}

}

}

return matrix;

}

1. Турнирная сортировка.

Код:

public static int[][] tournament(int[][] matrix)

{

int[][] array = new int[matrix.length][matrix[0].length];

for(int i = 0; i < matrix.length; i++)

{

for(int j = 0; j < matrix[i].length; j++)

{

int changer;

for(int comparator = 1; comparator < matrix[i].length; comparator \*= 2)

{

for(int k = 0; k\*comparator\*2 < matrix[i].length; k++)

{

if(k\*comparator\*2 + comparator < matrix[i].length)

{

if(matrix[i][k\*comparator\*2 + comparator] < matrix[i][k\*comparator\*2])

{

changer = matrix[i][k\*comparator\*2 + comparator];

matrix[i][k\*comparator\*2 + comparator] = matrix[i][k\*comparator\*2];

matrix[i][k\*comparator\*2] = changer;

}

}

}

}

array[i][j] = matrix[i][0];

matrix[i][0] = Integer.MAX\_VALUE;

}

}

System.out.println(Arrays.deepToString(array));

return array;

}

1. Быстрая сортировка.

Код:

public static int[][] quick(int[][] matrix)

{

for(int i = 0; i < matrix.length; i++)

{

matrix = hoar(matrix, i, 0, matrix[i].length-1);

}

System.out.println(Arrays.deepToString(matrix));

return matrix;

}

public static int[][] hoar(int[][] matrix, int i, int l, int r)

{

int pivot = matrix[i][l], j = l, k = r;

while(j < k)

{

while(matrix[i][k] >= pivot && j < k)

{

k--;

}

if(j != k)

{

matrix[i][j] = matrix[i][k];

j++;

}

while(matrix[i][j] <= pivot && j < k)

{

j++;

}

if(j != k)

{

matrix[i][k] = matrix[i][j];

k--;

}

}

matrix[i][j] = pivot;

if(l < j)

{

matrix = hoar(matrix, i, l, j-1);

}

if(r > j)

{

matrix = hoar(matrix, i, j+1, r);

}

return matrix;

}

1. Пирамидальная сортировка.

public static int[][] heap(int[][] matrix)

{

for(int i = 0; i < matrix.length; i++)

{

for(int j = matrix[i].length-1; j > 0; j--)

{

int k = 0, summ = 1, changer;

while(summ < j+1)

{

summ += 2;

if(summ < j+1)

{

k++;

}

}

//System.out.println(k);

for(int index = k; index >= 0; index--)

{

if(matrix[i][index] < matrix[i][2\*index+1])

{

changer = matrix[i][index];

matrix[i][index] = matrix[i][2\*index+1];

matrix[i][2\*index+1] = changer;

}

if(2\*index + 2 < j+1)

{

if(matrix[i][index] < matrix[i][2\*index+2])

{

changer = matrix[i][index];

matrix[i][index] = matrix[i][2\*index+2];

matrix[i][2\*index+2] = changer;

}

}

}

changer = matrix[i][0];

matrix[i][0] = matrix[i][j];

matrix[i][j] = changer;

}

}

System.out.println(Arrays.deepToString(matrix));

return matrix;

}