Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Лабораторная работа №1 по дисциплине

«Структуры и алгоритмы обработки данных»

Выполнил:

студент группы БВТ1902

Клюев А.П.

Введение

Цель данной лабораторной – получить знания и навыки написав

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 соответствии

с заданием. Оценить время работы каждого алгоритма сортировки и сравнить

его со временем стандартной функции сортировки. Испытания проводить на

сгенерированных матрицах.

Методы:

1. Выбором

2. Вставкой

3. Обменом

4. Шелла

5. Турнирная

6. Быстрая сортировка

7. Пирамидальная

Листинг программы

package com.sort;

import java.util.\*;

public class Sort {

public static int[][] create(int m,int n,int minl,int maxl){

final Random random = new Random();

int[][] a = new int[m][n];

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

for (int j = 0;j<a[0].length; j++){

a[i][j]= random.nextInt(maxl+Math.abs(minl))+minl;

}

}

return a;

}

public static void main(String[] args) {

var c = create(7,7,-250,1012);

matout(c);

System.out.println();

System.out.println();

matout(selectionSort(c));

System.out.println();

System.out.println();

matout(sortinsert(c));

System.out.println();

System.out.println();

matout(bubblesort(c));

System.out.println();

System.out.println();

matout(Shellsort(c));

System.out.println();

System.out.println();

matout(Quicksort(c));

System.out.println();

System.out.println();

matout(Heapsort(c));

System.out.println();

System.out.println();

matout(Standard(c));

}

public static void matout(int[][] a){

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

for (int j = 0; j < a[0].length; j++) {

System.out.print(" " + a[i][j] + " ");

}

System.out.println();

}

}

public static int[] tom(int[][] a){

int ch = 0;

int[] b = new int[a.length\*a[0].length];

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

for (int j = 0;j<a[0].length; j++){

b[ch]=a[i][j];

ch++;

}

}

return b;

}

public static List<Integer> tol(int[][] a){

List<Integer> b = new ArrayList<Integer>();

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

for (int j = 0;j<a[0].length; j++){

b.add(a[i][j]);

}

}

return b;

}

public static int[][] selectionSort(int[][] a){

int ch = 0;

int min = Integer.MAX\_VALUE;

int k = 0;

int ind = 0;

long t1= System.currentTimeMillis();

int[] b = tom(a);

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

for (int j = i;j<b.length; j++){

if (min>b[j]) {min=b[j];ind = j;}

}

k=b[i];b[i]=min; b[ind] = k;

min = Integer.MAX\_VALUE;

}

ch = 0;

int[][] d= new int[a.length][a[0].length];

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

for (int j = 0;j<d[0].length; j++){

d[i][j]=b[ch++];

}

}

long t2= System.currentTimeMillis();

System.out.println("Selection sort = " + Long.toString((t2-t1)));

return d;

}

public static int[][] sortinsert(int[][] a){

int k;

long t1= System.currentTimeMillis();

int[] b=tom(a);

for (int i = 1; i < b.length; i++) {

k=b[i];

for (int j = i-1; j >= 0 ; j--) {

if(k<b[j]){

b[j+1]=b[j];

if(j==0)b[j]=k;

}else{

b[j+1]=k;

break;

}

}

}

int ch = 0;

int[][] d= new int[a.length][a[0].length];

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

for (int j = 0;j<d[0].length; j++){

d[i][j]=b[ch];

ch++;

}

}

long t2= System.currentTimeMillis();

System.out.println("Insertion sort = " + Long.toString(t2-t1));

return d;

}

public static int[][] bubblesort(int[][] a){

long t1= System.currentTimeMillis();

int[] b = tom(a);

int k=0;

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

for (int j = 1; j < b.length-i; j++) {

if(b[j]<b[j-1]){k=b[j];b[j]=b[j-1];b[j-1]=k;}

}

}

int ch = 0;

int[][] d= new int[a.length][a[0].length];

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

for (int j = 0;j<d[0].length; j++){

d[i][j]=b[ch];

ch++;

}

}

long t2= System.currentTimeMillis();

System.out.println("bubble sort = " + Long.toString(t2-t1));

return d;

}

public static int[][] Shellsort(int[][] a){

long t1= System.currentTimeMillis();

int[] b = tom(a);

int k;

int n = b.length;

for (int step = n / 2; step > 0; step /= 2) {

for (int i = step; i < n; i++) {

for (int j = i - step; j >= 0 && b[j] > b[j + step] ; j -= step) {

int x = b[j];

b[j] = b[j + step];

b[j + step] = x;

}

}

}

int ch = 0;

int[][] d= new int[a.length][a[0].length];

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

for (int j = 0;j<d[0].length; j++){

d[i][j]=b[ch++];

}

}

long t2= System.currentTimeMillis();

System.out.println("Shell sort = " + Long.toString(t2-t1));

return d;

}

public static int[][] Quicksort(int[][] matrix){

long t1 = System.currentTimeMillis();

List<Integer> list = tol(matrix);

list = sort(list);

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

int ch = 0;

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

for (int j = 0;j < result[0].length; j++) {

result[i][j] = list.get(ch);

ch++;

}

}

long t2= System.currentTimeMillis();

System.out.println("Quick sort = " + Long.toString(t2-t1));

return result;

}

private static List<Integer> concat(List<Integer> left,int p,List<Integer> right){

List<Integer> result = new ArrayList<Integer>();

for (int i = 0; i < left.size(); i++) {

result.add(left.get(i));

}

result.add(p);

for (int i = 0; i < right.size(); i++) {

result.add(right.get(i));

}

return result;

}

private static List<Integer> sort(List<Integer> array){

if (array.size() < 2) return array;

int pivot = array.get(array.size()-1);

List<Integer> leftlist = new ArrayList<Integer>();

List<Integer> rightlist = new ArrayList<Integer>();

for (int i = 0;i < array.size()-1; i++) {

if (array.get(i) > pivot) {

rightlist.add(array.get(i));

} else {

leftlist.add(array.get(i));

}

}

return concat(sort(leftlist),pivot,sort(rightlist));

}

private static int[][] Heapsort(int[][] matrix){

int[] mas = tom(matrix);

long t1 = System.currentTimeMillis();

int n = mas.length;

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

for (int i = n / 2 - 1; i >= 0; i--)

heapify(mas, n, i);

for (int i = n - 1; i > 0; i--) {

int temp = mas[0];

mas[0] = mas[i];

mas[i] = temp;

heapify(mas, i, 0);

}

int ch = 0;

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

for (int j = 0;j < result[0].length; j++) {

result[i][j] = mas[ch++];

}

}

long t2= System.currentTimeMillis();

System.out.println("Heap sort = " + Long.toString(t2-t1));

return result;

}

private static void heapify(int[] mas, int n, int i){

int largest = i;

int l = 2\*i + 1;

int r = 2\*i + 2;

if (l < n && mas[l] > mas[largest])

largest = l;

if (r < n && mas[r] > mas[largest])

largest = r;

if (largest != i)

{

int swap = mas[i];

mas[i] = mas[largest];

mas[largest] = swap;

heapify(mas, n, largest);

}

}

private static int[][] Standard(int[][] matrix){

long t1 = System.currentTimeMillis();

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

int[] mas = tom(matrix);

Arrays.sort(mas);

int ch = 0;

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

for (int j = 0;j < result[0].length; j++) {

result[i][j] = mas[ch++];

}

}

long t2= System.currentTimeMillis();

System.out.println("Standart sort = " + Long.toString(t2-t1));

return result;

}

}

Вывод

В результате выполненной лабораторной работы я реализовал различные методы сортировки чисел для сгенерированной матрицы на java.