Министерство цифрового развития, связи и массовых коммуникаций
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
«Методы сортировки»
по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БВТ1902
Лапин Виктор Андреевич

Проверил: Кутейников Иван Алексеевич

Москва
2021

Оглавление

1. ВВЕДЕНИЕ

Цель данной лабораторной работы – создать генератор матриц, принимающий заданные параметры, изучить и реализовать различные алгоритмы сортировки, и применить их к сгенерированным матрицам.

2. ПОСТАНОВКА ЗАДАЧИ

Задание 1.

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.

Задание 2.

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

3. ЛИСТИНГ ПРОГРАММЫ

```
1   import java.util.*;
2
3   public class Main {
4     public static void main(String[] args) {
5       int[] sorted;
6       int m, n, minLimit, maxLimit;
7       Scanner sc = new Scanner(System.in);
8       System.out.println("m:");
9       try {
10        m = sc.nextInt();
11      } catch (InputMismatchException e) {
12        m = 50;
13        sc = new Scanner(System.in);
14      }
15      System.out.println("n:");
16      try {
17        n = sc.nextInt();
18      } catch (InputMismatchException e) {
19        n = 50;
20        sc = new Scanner(System.in);
21      }
22      System.out.println("minLimit:");
23      try {
24        minLimit = sc.nextInt();
25      } catch (InputMismatchException e) {
26        minLimit = -250;
27        sc = new Scanner(System.in);
28      }
29      System.out.println("maxLimit:");
30      try {
31        maxLimit = sc.nextInt();
32      } catch (InputMismatchException e) {
33        maxLimit = 1000 + 14;
34      }
35
36      int[][] array = matrix(m, n, minLimit, maxLimit);
37
38      System.out.println("\nOriginal:");
39      for (int j = 0; j < array.length; j++) {
40        sorted = array[j];
41        for (int i = 0; i < sorted.length; i++) {
42          System.out.print(sorted[i] + " ");
43        }
44        System.out.println("");
45      }
46
47      System.out.println("\nSelection sort:");
48      for (int j = 0; j < array.length; j++) {
49        sorted = selectionSort(array[j]);
```

```
50          for (int i = 0; i < sorted.length; i++) {
51            System.out.print(sorted[i] + " ");
52          }
53          System.out.println("");
54        }
55
56        System.out.println("\nInsertion sort:");
57        for (int j = 0; j < array.length; j++) {
58          sorted = insertionSort(array[j]);
59          for (int i = 0; i < sorted.length; i++) {
60            System.out.print(sorted[i] + " ");
61          }
62          System.out.println("");
63        }
64
65        System.out.println("\nBubble sort:");
66        for (int j = 0; j < array.length; j++) {
67          sorted = bubbleSort(array[j]);
68          for (int i = 0; i < sorted.length; i++) {
69            System.out.print(sorted[i] + " ");
70          }
71          System.out.println("");
72        }
73
74        System.out.println("\nShell sort:");
75        for (int j = 0; j < array.length; j++) {
76          sorted = ShellSort(array[j]);
77          for (int i = 0; i < sorted.length; i++) {
78            System.out.print(sorted[i] + " ");
79          }
80          System.out.println("");
81        }
82
83        System.out.println("\nTournament sort:");
84        for (int j = 0; j < array.length; j++) {
85          sorted = tournamentSort(array[j]);
86          for (int i = 0; i < sorted.length; i++) {
87            System.out.print(sorted[i] + " ");
88          }
89          System.out.println("");
90        }
91
92        System.out.println("\nQuick sort:");
93        for (int j = 0; j < array.length; j++) {
94          sorted = quickSort(array[j]);
95          for (int i = 0; i < sorted.length; i++) {
96            System.out.print(sorted[i] + " ");
97          }
98          System.out.println("");
99        }
100
101       System.out.println("\nHeap sort:");
```

```
102       for (int j = 0; j < array.length; j++) {
103         sorted = heapSort(array[j]);
104         for (int i = 0; i < sorted.length; i++) {
105           System.out.print(sorted[i] + " ");
106         }
107         System.out.println("");
108       }
109     }
110
111     public static int[][] matrix(int m, int n, int minLimit,
                                     int maxLimit) {
112       int[][] matr = new int[n][m];
113       Random random = new Random();
114       for (int i = 0; i < n; i++) {
115         for (int j = 0; j < m; j++) {
116           matr[i][j] = random.nextInt(maxLimit - minLimit) + minLimit;
117         }
118       }
119       return matr;
120     }
121
122     public static int[] selectionSort(int[] arr) {
123       int temp;
124       for (int pos = 0; pos < arr.length - 1; pos++) {
125         int min = arr[pos];
126         int index = pos;
127         for (int i = pos + 1; i < arr.length; i++) {
128           if (arr[i] < min) {
129             index = i;
130             min = arr[i];
131           }
132         }
133         temp = arr[pos];
134         arr[pos] = arr[index];
135         arr[index] = temp;
136       }
137       return arr;
138     }
139
140     public static int[] insertionSort(int[] arr) {
141       int temp;
142       for (int i = 0; i < arr.length - 1; i++) {
143         for (int j = i + 1; j > 0; j--) {
144           if (arr[j] > arr[j - 1]) {
145             break;
146           } else {
147             temp = arr[j];
148             arr[j] = arr[j - 1];
149             arr[j - 1] = temp;
150           }
151         }
152       }
```

```
153       return arr;
154     }
155
156     public static int[] bubbleSort(int[] arr) {
157       int temp;
158       for (int i = arr.length - 1; i >= 0; i--) {
159         for (int j = 0; j < i; j++) {
160           if (arr[j] > arr[j + 1]) {
161             temp = arr[j];
162             arr[j] = arr[j + 1];
163             arr[j + 1] = temp;
164           }
165         }
166       }
167       return arr;
168     }
169
170     public static int[] ShellSort(int[] arr) {
171       int temp;
172       int d = arr.length;
173       boolean order = false;
174       while (!order || d > 1) {
175         order = true;
176         d = (int) Math.ceil((double) d / 2);
177         for (int j = 0; j < arr.length - d; j++) {
178           if (arr[j] > arr[j + d]) {
179             temp = arr[j];
180             arr[j] = arr[j + d];
181             arr[j + d] = temp;
182             order = false;
183           }
184         }
185       }
186       return arr;
187     }
188
189     public static int[] tournamentSort(int[] arr) {
190       int num = 1;
191       int len = arr.length;
192       while (len > 1) {
193         len = (int) Math.ceil((float) len / 2);
194         num++;
195       }
196       int[][] tree = new int[num][];
197       len = arr.length;
198       for (int i = 0; i < num; i++) {
199         if (len != 1 && len % 2 != 0) {
200           len++;
201           tree[i] = new int[len];
202           tree[i][tree[i].length - 1] = Integer.MAX_VALUE;
203         } else {
204           tree[i] = new int[len];
```

```
205         }
206         len = (int) Math.ceil((float) len / 2);
207       }
208       System.arraycopy(arr, 0, tree[0], 0, arr.length);
209       num = Integer.MAX_VALUE;
210       for (int k = 0; k < arr.length; k++) {
211         for (int i = 0; i < tree.length - 1; i++) {
212           for (int j = 0; j < tree[i].length; j += 2) {
213             if (i == 0 && tree[i][j] == num) {
214               tree[0][j] = Integer.MAX_VALUE;
215               num = Integer.MAX_VALUE;
216             }
217             if (i == 0 && tree[i][j + 1] == num) {
218               tree[0][j + 1] = Integer.MAX_VALUE;
219               num = Integer.MAX_VALUE;
220             }
221             if (tree[i][j] > tree[i][j + 1]) {
222               tree[i + 1][j / 2] = tree[i][j + 1];
223             } else {
224               tree[i + 1][j / 2] = tree[i][j];
225             }
226           }
227         }
228         num = tree[tree.length - 1][0];
229         arr[k] = num;
230       }
231       return arr;
232     }
233
234     public static int[] quickSort(int[] arr) {
235       if (arr.length <= 1) {
236         return arr;
237       }
238       int index = (int) Math.floor((float) arr.length / 2);
239       int temp;
240       for (int i = 0; i < index; i++) {
241         if (arr[i] > arr[index]) {
242           temp = arr[i];
243           for (int j = i + 1; j <= index; j++) {
244             arr[j - 1] = arr[j];
245           }
246           arr[index] = temp;
247           index--;
248           i--;
249         }
250       }
251       for (int i = arr.length - 1; i > index; i--) {
252         if (arr[i] < arr[index]) {
253           temp = arr[i];
254           for (int j = i - 1; j >= index; j--) {
255             arr[j + 1] = arr[j];
256           }
```

```
257           arr[index] = temp;
258           index++;
259           i++;
260         }
261       }
262       int[] left = new int[index];
263       int[] right = new int[arr.length - index - 1];
264       System.arraycopy(arr, 0, left, 0, index);
265       System.arraycopy(arr, index + 1, right, 0,
                           arr.length - index - 1);
266       System.arraycopy(quickSort(left), 0, arr, 0, index);
267       System.arraycopy(quickSort(right), 0, arr, index + 1,
                           arr.length - index - 1);
268       return arr;
269     }
270
271     public static int[] heapSort(int[] arr) {
272       int temp;
273       int state = arr.length - 1;
274       while (state > 1) {
275         for (int i = state; i > 0; i--) {
276           if (arr[i] > arr[(i - 2 + (i % 2)) / 2]) {
277             temp = arr[i];
278             arr[i] = arr[(i - 2 + (i % 2)) / 2];
279             arr[(i - 2 + (i % 2)) / 2] = temp;
280           }
281         }
282         temp = arr[0];
283         arr[0] = arr[state];
284         arr[state] = temp;
285         state--;
286       }
287       return arr;
288     }
289   }
```

4. ВЫВОД

В результате выполнения данной лабораторной работы на языке программирования Java создан генератор матриц, принимающий минимальное и максимальное значение элементов и их количество. Изучены и реализованы различные алгоритмы сортировки, такие как сортировка выбором, вставкой, обменом, Шелла, турнирная, быстрая, пирамидальная, которые применены к сгенерированным матрицам.