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 № 2
«Методы поиска»
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 – изучить и реализовать различные алгоритмы поиска и рехеширования, применить их к различным наборам данных.

2. ПОСТАНОВКА ЗАДАЧИ

Задание 1.

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных. Для всех вариантов добавить реализацию добавления, поиска и удаления элементов. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

Методы:

Бинарный поиск	Бинарное дерево	Фибоначчиев	Интерполяционный

Задание 2.

Реализовать методы рехеширования.

Методы:

Простое рехеширование	Метод цепочек

Задание 3.

Расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого». Подразумевается, что ферзь бьёт все клетки, расположенные по вертикалям, горизонталям и обеим диагоналям.

Написать программу, которая находит хотя бы один способ решения задач.

3. ЛИСТИНГ ПРОГРАММЫ

Класс Main

```
1    import java.util.Arrays;
2    import java.util.LinkedList;
3    import java.util.Scanner;
4
5    public class Main {
6      public static void main(String[] args) {
7        int[] intArr = Matrix.matrix(1000, 1, 0, 100000)[0];
8        BinaryTree tree = new BinaryTree();
9        for (int i = 0; i < intArr.length; i++) {
10         tree.addElement(intArr[i]);
11       }
12       Matrix.quickSort(intArr);
13
14       System.out.println("\n\u001B[47m\u001B[30m
                              Исходный массив:\u001B[0m");
15       for (int i = 0; i < intArr.length; i++) {
16         System.out.print(intArr[i] + " ");
17       }
18
19       System.out.println("\n\nДобавление элемента");
20       int toAdd = 1234;
21       intArr = addToArray(intArr, toAdd);
22       tree.addElement(toAdd);
23       System.out.print("Элемент " + toAdd + " добавлен в массив ");
24
25       System.out.println("\n\n\u001B[47m\u001B[30m
                             Бинарный поиск:\u001B[0m");
26       int toFind = intArr[753];
27       long time = System.nanoTime();
28       int found = binarySearch(intArr, toFind);
29       time = System.nanoTime() - time;
30       if (found != -1) {
31         System.out.print("Элемент " + toFind + " найден на позиции " +
found);
32       } else {
33         System.out.print("Элемент " + toFind + " не найден");
34       }
35       System.out.print("\n" + time + " ns");
36
37       System.out.println("\n\n\u001B[47m\u001B[30m
                             Бинарное дерево:\u001B[0m");
38       time = System.nanoTime();
39       tree.searchElement(toFind);
40       time = System.nanoTime() - time;
41       System.out.print("Путь к элементу " + toFind + ": " +
tree.pathToElement(toFind));
```

```
42        System.out.print("\n" + time + " ns");
43
44        System.out.println("\n\n\u001B[47m\u001B[30m
                             Фибоначчиев поиск:\u001B[0m");
45        time = System.nanoTime();
46        found = FibonacciSearch(intArr, toFind);
47        time = System.nanoTime() - time;
48        if (found != -1) {
49          System.out.print("Элемент " + toFind + " найден на позиции " +
                             found);
50        } else {
51          System.out.print("Элемент " + toFind + " не найден");
52        }
53        System.out.print("\n" + time + " ns");
54
55        System.out.println("\n\n\u001B[47m\u001B[30m
                            Интерполяционный поиск:\u001B[0m");
56        time = System.nanoTime();
57        found = interpolationSearch(intArr, toFind);
58        time = System.nanoTime() - time;
59        if (found != -1) {
60          System.out.print("Элемент " + toFind + " найден на позиции " +
                             found);
61        } else {
62          System.out.print("Элемент " + toFind + " не найден");
63        }
64        System.out.print("\n" + time + " ns");
65
66        String[] arr = new String[]{"apple", "banana", "strawberry",
                                      "lemon", "cherry", "pear",
                                      "watermelon", "orange", "pineapple"};
67
68        System.out.println("\n\n\u001B[47m\u001B[30m
                             Простое рехеширование:\u001B[0m");
69        String[] hashTable = new String[9];
70        Arrays.fill(hashTable, "");
71
72        System.out.println("Результат первичного хеширования:");
73        for (int i = 0; i < arr.length; i++) {
74          System.out.println("\"" + arr[i] + "\": " +
                               hashFunction(arr[i], 0));
75        }
76
77        for (int i = 0; i < arr.length; i++) {
78          for (int j = 0; j < hashTable.length; j++) {
79            int hash = hashFunction(arr[i], j);
80            if (hashTable[hash].equals("")) {
81              hashTable[hash] = arr[i];
82              break;
83            }
84            if (j == hashTable.length - 1) {
```

```
85              System.out.println("Невозможно записать элемент \"" +
                                   arr[i] + "\" в хеш-таблицу!");
86            }
87          }
88        }
89
90        System.out.println("\nХеш-таблица:");
91        for (int i = 0; i < hashTable.length; i++) {
92          System.out.println(i + ": \"" + hashTable[i] + "\"");
93        }
94
95        System.out.println("\nПоиск элемента");
96        String element = "strawberry";
97        for (int j = 0; j < hashTable.length; j++) {
98          int hash = hashFunction(element, j);
99          if (hashTable[hash].equals(element)) {
100           System.out.println("Элемент \"" + element +
                                 "\" найден с индексом " + hash);
101           break;
102         }
103         if (j == hashTable.length - 1) {
104           System.out.println("Элемент не найден");
105         }
106       }
107
108       System.out.println("\nУдаление элемента");
109       element = "watermelon";
110       for (int j = 0; j < hashTable.length; j++) {
111         int hash = hashFunction(element, j);
112         if (hashTable[hash].equals(element)) {
113           hashTable[hash] = "";
114           System.out.println("Элемент \"" + element +
                                 "\" удалён по индексу " + hash);
115           break;
116         }
117         if (j == hashTable.length - 1) {
118           System.out.println("Элемент не найден");
119         }
120       }
121
122       System.out.println("\nДобавление элемента");
123       element = "raspberry";
124       for (int j = 0; j < hashTable.length; j++) {
125         int hash = hashFunction(element, j);
126         if (hashTable[hash].equals("")) {
127           hashTable[hash] = element;
128           System.out.println("Элемент \"" + element +
                                 "\" добавлен по индексу " + hash);
129           break;
130         }
131         if (j == hashTable.length - 1) {
```

```
132         System.out.println("Невозможно записать элемент \"" +
                               element + "\" в хеш-таблицу!");
133       }
134     }
135
136     System.out.print("\n\u001B[47m\u001B[30m
                         Метод цепочек:\u001B[0m");
137     LinkedList < String > [] list = new LinkedList[9];
138
139     for (int i = 0; i < arr.length; i++) {
140       int hash = hashFunction(arr[i], 0);
141       if (list[hash] == null) {
142         list[hash] = new LinkedList < > ();
143       }
144       list[hash].add(arr[i]);
145     }
146
147     for (int i = 0; i < list.length; i++) {
148       System.out.print("\n" + i + ": ");
149       if (list[i] != null) {
150         for (int j = 0; j < list[i].size(); j++) {
151           System.out.print(list[i].get(j) + " -> ");
152         }
153       }
154       System.out.print("null");
155     }
156
157     System.out.println("\n\nПоиск элемента");
158     element = "pineapple";
159     int hash = hashFunction(element, 0);
160     if (list[hash] != null) {
161       System.out.print(hash + ": ");
162       for (int j = 0; j < list[hash].size(); j++) {
163         String str = list[hash].get(j);
164         System.out.print(str + " -> ");
165         if (str.equals(element)) {
166           System.out.println("Элемент найден.");
167           break;
168         }
169       }
170     } else {
171       System.out.println("Элемент не найден.");
172     }
173
174     System.out.println("\nУдаление элемента");
175     element = "pear";
176     hash = hashFunction(element, 0);
177     if (list[hash] != null) {
178       System.out.print(hash + ": ");
179       for (int j = 0; j < list[hash].size(); j++) {
180         String str = list[hash].get(j);
181         System.out.print(str + " -> ");
```

```
182           if (str.equals(element)) {
183             list[hash].remove(j);
184             System.out.println("Элемент удалён.");
185             break;
186           }
187         }
188       } else {
189         System.out.println("Элемент не найден.");
190       }
191
192       System.out.println("\nДобавление элемента");
193       element = "raspberry";
194       hash = hashFunction(element, 0);
195       if (list[hash] == null) {
196         list[hash] = new LinkedList < > ();
197       }
198       list[hash].add(element);
199       System.out.println("Элемент " + element + " добавлен.");
200
201       System.out.print("\n\u001B[47m\u001B[30m
                          Задача о восьми ферзях:\u001B[0m\n");
202       int[][] field = eightQueens();
203       for (int i = 0; i < 8; i++) {
204         for (int j = 0; j < 8; j++) {
205           System.out.print(field[i][j] + " ");
206         }
207         System.out.println();
208       }
209     }
210
211     public static int[] addToArray(int[] arr, int elem) {
212       int[] out = new int[arr.length + 1];
213       System.arraycopy(arr, 0, out, 0, arr.length);
214       out[arr.length] = elem;
215       return Matrix.quickSort(out);
216     }
217
218     public static int binarySearch(int[] sortedArray, int elem) {
219       int low = 0;
220       int high = sortedArray.length;
221       int mid;
222       while (low < high) {
223         mid = (low + high) / 2;
224         if (sortedArray[mid] == elem) {
225           return mid;
226         } else if (sortedArray[mid] > elem) {
227           high = mid;
228         } else if (sortedArray[mid] < elem) {
229           low = mid + 1;
230         }
231       }
232       return -1;
```

```
233     }
234
235     public static int FibonacciSearch(int[] sortedArray, int elem) {
236       int low = 0;
237       int high = sortedArray.length - 1;
238       int f0 = 0;
239       int f1 = 1;
240       int f2 = 1;
241       if (elem < sortedArray[low] || elem > sortedArray[high]) {
242         return -1;
243       }
244       if (sortedArray[0] == elem) {
245         return sortedArray[0];
246       }
247       while (low < high) {
248         if (sortedArray[low + f2] == elem) {
249           return low + f2;
250         }
251         f0 = f1;
252         f1 = f2;
253         f2 = f0 + f1;
254         if (f2 > sortedArray.length) {
255           f2 = sortedArray.length - 1;
256         }
257         if (elem < sortedArray[f2]) {
258           low += f1 - 1;
259           f1 = 0;
260           f2 = 1;
261         }
262       }
263       return -1;
264     }
265
266     public static int interpolationSearch(int[] sortedArray, int elem){
267       int i = 0;
268       int j = sortedArray.length - 1;
269       int d;
270       while (i < j) {
271         d = i + ((j - i) * (elem - sortedArray[i])) /
              (sortedArray[j] - sortedArray[i]);
272         if (sortedArray[d] == elem) {
273           return d;
274         } else if (sortedArray[d] > elem) {
275           j = d - 1;
276         } else if (sortedArray[d] < elem) {
277           i = d + 1;
278         }
279       }
280       return -1;
281     }
282
283     public static int hashFunction(String str, int pi) {
```

```
284       return (str.charAt(0) + str.charAt(str.length() - 1) +
                  str.charAt((str.length() - 1) / 2) + pi) % 9;
285     }
286
287     public static int[][] eightQueens() {
288       int[] field = new int[64];
289       field = placeQueen(field, 0);
290       int[][] out = new int[8][8];
291       for (int i = 0; i < 8; i++) {
292         for (int j = 0; j < 8; j++) {
293           if (field[i * 8 + j] == 1) {
294             out[i][j] = 1;
295           } else {
296             out[i][j] = 0;
297           }
298         }
299       }
300       return out;
301     }
302
303     public static int[] placeQueen(int[] fieldc, int line) {
304       for (int i = 0; i < 8; i++) {
305         int[] field = fieldc.clone();
306         if (field[line * 8 + i] == 0) {
307           for (int j = 0; j < 8; j++) {
308             field[line * 8 + j] = 2;
309             field[j * 8 + i] = 2;
310             if ((line - i + j >= 0) && (line - i + j < 8)) {
311               field[(line - i + j) * 8 + j] = 2;
312             }
313             if ((line + i - j >= 0) && (line + i - j < 8)) {
314               field[(line + i - j) * 8 + j] = 2;
315             }
316           }
317           field[line * 8 + i] = 1;
318           if (line == 7) {
319             return field;
320           }
321           if (placeQueen(field, line + 1) != null) {
322             return placeQueen(field, line + 1);
323           }
324         }
325       }
326       return null;
327     }
328   }
```

Класс BinaryTree

```
1   public class BinaryTree {
2     float root = -1;
3     BinaryTree leftTree;
4     BinaryTree rightTree;
5
6     float searchElement(float element) {
7       if (root == -1) {
8         return -1;
9       } else if (element == root) {
10        return root;
11      } else if (element > root) {
12        if (rightTree == null) {
13          return -1;
14        }
15        return rightTree.searchElement(element);
16      } else if (element < root) {
17        if (leftTree == null) {
18          return -1;
19        }
20        return leftTree.searchElement(element);
21      }
22      return -1;
23    }
24
25    String pathToElement(float element) {
26      String path = "";
27      if (root == -1) {
28        return " Not found";
29      } else if (element == root) {
30        return "";
31      } else if (element > root) {
32        if (rightTree == null) {
33          return " Not found";
34        }
35        path += "R" + rightTree.pathToElement(element);
36      } else if (element < root) {
37        if (leftTree == null) {
38          return " Not found";
39        }
40        path += "L" + leftTree.pathToElement(element);
41      }
42      return path;
43    }
44
45    void addElement(float element) {
46      if (root == -1) {
47        root = element;
48      } else if (element == root) {
49        // Элемент уже есть
50      } else if (element > root) {
```

```
51          if (rightTree == null) {
52            rightTree = new BinaryTree();
53          }
54          rightTree.addElement(element);
55        } else if (element < root) {
56          if (leftTree == null) {
57            leftTree = new BinaryTree();
58          }
59          leftTree.addElement(element);
60        }
61      }
62    }
```

4. ВЫВОД

В результате выполнения данной лабораторной работы изучены и реализованы различные алгоритмы поиска, такие как бинарный поиск, бинарное дерево, Фибоначчиев, интерполяционный, и методы рехеширования, такие как простое рехеширование и метод цепочек. Все методы применены к различным наборам данных. Также реализован алгоритм, решающий задачу о расстановке восьми ферзей на шахматном поле.