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 № 3
«Методы поиска подстроки в строке»
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 подстроки в строке, применить их к различным наборам данных.

2. ПОСТАНОВКА ЗАДАЧИ

Задание 1.

Реализовать методы поиска подстроки в строке. Добавить возможность ввода строки и подстроки с клавиатуры. Предусмотреть возможность существования пробела. Реализовать возможность выбора опции чувствительности или нечувствительности к регистру. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования. Алгоритмы: Кнута-Морриса-Пратта, упрощённый Бойера-Мура

Задание 2.

Игра в 15, пятнашки, такен — популярная головоломка, придуманная в 1878 году Ноем Чепмэном. Она представляет собой набор одинаковых квадратных костяшек с нанесёнными числами, заключённых в квадратную коробку. Длина стороны коробки в четыре раза больше длины стороны костяшек для набора из 15 элементов, соответственно в коробке остаётся незаполненным одно квадратное поле. Цель игры — перемещая костяшки по коробке, добиться упорядочивания их по номерам, желательно сделав как можно меньше перемещений.

Задача: написать программу, определяющую, является ли данное расположение «решаемым», то есть можно ли из него за конечное число шагов перейти к правильному. Если это возможно, то необходимо найти хотя бы одно решение - последовательность движений, после которой числа будут расположены в правильном порядке. Входные данные: массив чисел, представляющий собой расстановку в порядке «слева направо, сверху вниз». Число 0 обозначает пустое поле. Выходные данные: если решения нет, то функция должна вернуть пустой массив []. Если решение есть, то необходимо представить решение — для каждого шага записывается номер передвигаемого на данном шаге элемента.

3. ЛИСТИНГ ПРОГРАММЫ

Класс Main

```
1   import java.util.ArrayList;
2   import java.util.Arrays;
3   import java.util.Random;
4   import java.util.Scanner;
5
6   public class Main {
7     public static void main(String[] args) {
8       /*Scanner sc = new Scanner(System.in);
9       sc.useDelimiter("\n");
10      System.out.print("Введите строку: ");
11      String str = sc.next();
12      System.out.print("Введите подстроку: ");
13      String subStr = sc.next();
14      int[] res = KMP(str, subStr);*/
15      int[] res = KMP("aabaabaaaabaabaaab", "aabaa");
16      System.out.print("Алгоритм КМП: Найдено совпадений: " +
                         res.length + ", ");
17      if (res.length > 0) {
18        System.out.print("в позициях: ");
19      }
20      for (int i = 0; i < res.length; i++) {
21        System.out.print(res[i] + ", ");
22      }
23      System.out.print("\b\b.\n");
24
25      /*System.out.print("Введите строку: ");
26      str = sc.next();
27      System.out.print("Введите подстроку: ");
28      subStr = sc.next();
29      res = BoyerMoore(str, subStr);*/
30      res = BoyerMoore("Sed ut perspiciatis unde omnis iste natus error
                         sit voluptatem accusantium doloremque
                         laudantium, totam rem aperiam", "error");
31      System.out.print("\nАлгоритм Бойера-Мура: Найдено совпадений: " +
                         res.length + ", ");
32      if (res.length > 0) {
33        System.out.print("в позициях: ");
34      }
35      for (int i = 0; i < res.length; i++) {
36        System.out.print(res[i] + ", ");
37      }
38      System.out.print("\b\b.\n");
39
40      int[] arr = new int[] {1, 2, 3, 4, 5, 6, 7, 8, 13, 9, 11, 12, 10,
                               14, 15, 0};
41      Random rnd = new Random();
```

```
42        for (int k = 0; k < 2; k++) {
43          for (int i = 0; i < arr.length; i++) {
44            if (k == 0) {
45              break;
46            }
47            if (k == 1) {
48              arr = new int[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
                                 15, 14, 0};
49              break;
50            }
51            int index = rnd.nextInt(i + 1);
52            int a = arr[index];
53            arr[index] = arr[i];
54            arr[i] = a;
55          }
56          System.out.print("\nИсходное состояние: \n");
57          for (int i = 0; i < 4; i++) {
58            for (int j = 0; j < 4; j++) {
59              System.out.print(arr[i * 4 + j] + " ");
60            }
61            System.out.print("\n");
62          }
63          System.out.print("Решение: ");
64          res = solveFifteenPuzzle(arr);
65          for (int i = 0; i < res.length; i++) {
66            System.out.print(res[i] + ", ");
67          }
68          System.out.print("\b\b");
69          if (res.length == 0) {
70            System.out.print(" не существует");
71          }
72          System.out.print(".\n");
73        }
74      }
75
76      public static int[] KMP(String str, String subStr) {
77        str = subStr.concat("\0").concat(str);
78        int[] prefix = prefix(str);
79        int cnt = 0;
80        for (int i = 0; i < prefix.length; i++) {
81          if (prefix[i] == subStr.length()) {
82            cnt++;
83          }
84        }
85        int[] out = new int[cnt];
86        cnt = 0;
87        for (int i = 0; i < prefix.length; i++) {
88          if (prefix[i] == subStr.length()) {
89            out[cnt] = i - subStr.length() * 2;
90            cnt++;
91          }
92        }
```

```
93        return out;
94      }
95
96      public static int[] prefix(String str) {
97        char[] ch = str.toCharArray();
98        int[] prefix = new int[ch.length];
99        for (int i = 0; i < ch.length; i++) { // для каждого символа
100         int maxCnt = 0;
101         for (int j = 0; j < i; j++) { // длина от 0 до позиции символа
102           int cnt = 0;
103           for (int k = 0; k < j + 1; k++) { // нахождение совпадений
104             if (ch[k] == ch[i - j + k]) {
105               cnt++;
106             } else {
107               cnt = 0;
108               break;
109             }
110           }
111           if (cnt > maxCnt) {
112             maxCnt = cnt;
113           }
114         }
115         prefix[i] = maxCnt;
116       }
117       return prefix;
118     }
119
120     public static int[] BoyerMoore(String string, String subString) {
121       char[] str = string.toCharArray();
122       char[] subStr = subString.toCharArray();
123       int[] d = new int[256];
124       int[] subStringNumbers = new int[subStr.length];
125       Arrays.fill(d, subStr.length);
126       for (int i = subStr.length - 1; i >= 0; i--) {
127         if (d[subStr[i]] == subStr.length) {
128           subStringNumbers[i] = subStr.length - i - 1;
129           d[subStr[i]] = subStr.length - i - 1;
130         } else {
131           subStringNumbers[i] = d[subStr[i]];
132         }
133       }
134       int offset = 0;
135       int[] out = new int[0];
136       while (offset + subStr.length <= str.length) {
137         for (int j = subStr.length - 1; j >= 0; j--) {
138           if (str[j + offset] != subStr[j]) {
139             offset += d[str[j + offset]];
140             break;
141           }
142           if (j == 0) {
143             int[] temp = new int[out.length + 1];
144             System.arraycopy(out, 0, temp, 0, out.length);
```

```
145             out = temp;
146             out[out.length - 1] = j + offset;
147             //System.out.println("найден");
148             offset++;
149           }
150         }
151       }
152       return out;
153     }
154
155     public static int[] solveFifteenPuzzle(int[] arr) {
156       Fifteen fifteen = new Fifteen(arr);
157       return fifteen.solve();
158     }
159   }
```

Класс Fifteen

```
1    public class Fifteen {
2      private int[] cells;
3      private int[] solution = new int[0];
4
5      public Fifteen(int[] cells) {
6        if (cells.length != 16) {
7          throw new IllegalArgumentException("Field should contain 16
                                                 elements");
8        }
9        for (int i = 0; i < 16; i++) {
10         for (int j = 0; j < 16; j++) {
11           if (cells[i] == j) {
12             break;
13           }
14           if (j == 15) {
15             throw new IllegalArgumentException("Not enough numbers");
16           }
17         }
18       }
19       this.cells = cells;
20     }
21
22     public int[] solve() {
23       if (!isSolvable()) {
24         return solution;
25       }
26       if (!checkRow(1)) {
27         placeAt(1, 1, 1);
28         //System.out.println("1 PLACED");
29         placeAt(2, 1, 2);
30         //System.out.println("2 PLACED");
31         placeAt(4, 1, 3);
32         //System.out.println("4 PLACED");
33         placeAt(3, 2, 3);
34         if (getRow(4) == 1 && getCol(4) == 3 && getRow(3) == 2 &&
               getCol(3) == 3) {
35           if (getCol(0) != 4) {
36             while (getRow(0) < 3) {
37               moveZero(3);
38             }
39             while (getCol(0) < 4) {
40               moveZero(2);
41             }
42           }
43           while (getRow(0) > 1) {
44             moveZero(1);
45           }
46           moveZero(4);
47           moveZero(3);
48         }
```

```
49          //System.out.println("3 PLACED");
50        }
51
52        if (!checkRow(2)) {
53          placeAt(5, 2, 1);
54          //System.out.println("5 PLACED");
55          placeAt(6, 2, 2);
56          //System.out.println("6 PLACED");
57          placeAt(8, 2, 3);
58          //System.out.println("8 PLACED");
59          placeAt(7, 3, 3);
60          if (getRow(8) == 2 && getCol(8) == 3 && getRow(7) == 3 &&
                getCol(7) == 3) {
61            if (getCol(0) != 4) {
62              while (getRow(0) < 4) {
63                moveZero(3);
64              }
65              while (getCol(0) < 4) {
66                moveZero(2);
67              }
68            }
69            while (getRow(0) > 2) {
70              moveZero(1);
71            }
72            moveZero(4);
73            moveZero(3);
74          }
75          //System.out.println("7 PLACED");
76        }
77        if (!checkRow(3) && !checkRow(4)) {
78          placeAt(13, 3, 1);
79          //System.out.println("13 PLACED");
80          placeAt(9, 3, 2);
81          if (getRow(13) == 4 && getCol(13) == 1 && getRow(9) == 3 &&
                getCol(9) == 2) {
82            border(9);
83          } else {
84            if (getRow(0) == 3) {
85              moveZero(3);
86            }
87            while (getCol(0) > 1) {
88              moveZero(4);
89            }
90            moveZero(1);
91            moveZero(2);
92          }
93          //System.out.println("9 PLACED");
94          placeAt(14, 3, 2);
95          //System.out.println("14 PLACED");
96          placeAt(10, 3, 3);
97          if (getRow(14) == 4 && getCol(14) == 2 && getRow(10) == 3 &&
                getCol(10) == 3) {
```

```
98          border(10);
99        } else {
100         if (getRow(0) == 3) {
101           moveZero(3);
102         }
103         while (getCol(0) > 2) {
104           moveZero(4);
105         }
106         moveZero(1);
107         moveZero(2);
108       }
109       //System.out.println("10 PLACED");
110       while (!(getCol(11) == 3 && getRow(11) == 3 && getCol(12) == 4
                && getRow(12) == 3)) {
111         if (getRow(0) == 3) {
112           if (getCol(0) == 3) {
113             moveZero(2);
114           }
115           if (getCol(0) == 4) {
116             moveZero(3);
117           }
118         }
119         if (getRow(0) == 4) {
120           if (getCol(0) == 3) {
121             moveZero(1);
122           }
123           if (getCol(0) == 4) {
124             moveZero(4);
125           }
126         }
127       }
128       if (getCol(15) == 4) {
129         moveZero(2);
130       }
131     }
132     //System.out.println("SOLVED");
133     for (int i = 0; i < 15; i++) {
134       if (cells[i] != i + 1) {
135         System.out.println("NOT SOLVED");
136         System.exit(-1);
137       }
138     }
139     if (cells[15] != 0) {
140       System.out.println("NOT SOLVED");
141       System.exit(-1);
142     }
143     return solution;
144   }
145
146   public void placeAt(int num, int row, int col) {
147     // выставляем в нужный столбец
148     if (num == 3 && getRow(3) == 1) {
```

```
149         border(3);
150         return;
151       }
152       if (num == 7 && getRow(7) == 2) {
153         border(7);
154         return;
155       }
156       while (getCol(num) != col) {
157         if (getRow(num) < 4) {
158           while (getRow(num) + 1 != getRow(0)) {
159             if (getRow(0) <= getRow(num)) {
160               moveZero(3);
161             } else {
162               moveZero(1);
163             }
164           }
165           if (num == 3 && getRow(3) == 1) {
166             border(3);
167             return;
168           }
169           if (num == 7 && getRow(7) == 2) {
170             border(7);
171             return;
172           }
173           // теперь 0 находится в строке под элементом
174           if (getCol(num) > col) { //если элемент правее чем нужно
175             while (getCol(0) + 1 != getCol(num)) {
176               if (getCol(0) < getCol(num) - 1) {
177                 moveZero(2);
178               } else {
179                 moveZero(4);
180               }
181             }
182             moveZero(1);
183             moveZero(2);
184           }
185           if (getCol(num) < col) { //если элемент левее чем нужно
186             while (getCol(0) - 1 != getCol(num)) {
187               if (getCol(0) - 1 < getCol(num)) {
188                 moveZero(2);
189               } else {
190                 moveZero(4);
191               }
192             }
193             moveZero(1);
194             moveZero(4);
195           }
196         } else {
197           while (getRow(0) + 1 != getRow(num)) {
198             if (getRow(0) + 1 < getRow(num)) {
199               moveZero(3);
200             } else {
```

```
201             moveZero(1);
202           }
203         }
204         // теперь 0 находится в строке над элементом
205         if (getCol(num) > col) { //если элемент правее чем нужно
206           while (getCol(0) + 1 != getCol(num)) {
207             if (getCol(0) < getCol(num) - 1) {
208               moveZero(2);
209             } else {
210               moveZero(4);
211             }
212           }
213           moveZero(3);
214           moveZero(2);
215         }
216         if (getCol(num) < col) { //если элемент левее чем нужно
217           while (getCol(0) - 1 != getCol(num)) {
218             if (getCol(0) - 1 < getCol(num)) {
219               moveZero(2);
220             } else {
221               moveZero(4);
222             }
223           }
224           moveZero(3);
225           moveZero(4);
226         }
227       }
228     }
229     // ставим в нужную строку
230     while (getRow(num) != row) {
231       if (getRow(num) < 4) {
232         if (getCol(0) < getCol(num)) { // если 0 справа от элемента
                                           то обходим снизу
233           while (getRow(0) - 1 != getRow(num)) {
234             if (getRow(0) > getRow(num)) {
235               moveZero(1);
236             } else {
237               moveZero(3);
238             }
239           }
240         }
241       } else {
242         if (getCol(0) < getCol(num)) { // если 0 справа от элемента
                                           то обходим сверху
243           while (getRow(0) + 1 != getRow(num)) {
244             if (getRow(0) >= getRow(num)) {
245               moveZero(1);
246             } else {
247               moveZero(3);
248             }
249           }
250         }
```

```
251         }
252         while (getCol(0) - 1 != getCol(num)) {
253           if (getCol(0) > getCol(num)) {
254             moveZero(4);
255           } else {
256             moveZero(2);
257           }
258         }
259         while (getRow(0) + 1 != getRow(num)) {
260           if (getRow(0) < getRow(num)) {
261             moveZero(3);
262           } else {
263             moveZero(1);
264           }
265         }
266         moveZero(4);
267         moveZero(3);
268       }
269     }
270
271     public void border(int num) {
272       int[] moves = new int[] {1, 2, 3, 4, 3, 2, 1, 1, 4, 3, 2, 3, 4,
                                  1, 1, 2, 3};
273       int[] movesDown = new int[] {4, 1, 2, 2, 3, 4, 4, 1, 2, 3, 2, 1,
                                     4, 4, 3, 2};
274       //System.out.println("ОБРАБОТКА ГРАНИЧНОГО УСЛОВИЯ" + num);
275       if (num == 3) {
276         while (getRow(0) != 2) {
277           if (getRow(0) > 2) {
278             moveZero(1);
279           } else {
280             moveZero(3);
281           }
282         }
283         while (getCol(0) != 3) {
284           if (getCol(0) > 3) {
285             moveZero(4);
286           } else {
287             moveZero(2);
288           }
289         }
290         for (int i = 0; i < moves.length; i++) {
291           moveZero(moves[i]);
292         }
293       }
294
295       if (num == 7) {
296         while (getRow(0) != 3) {
297           if (getRow(0) > 3) {
298             moveZero(1);
299           } else {
300             moveZero(3);
```

```
301           }
302         }
303         while (getCol(0) != 3) {
304           if (getCol(0) > 3) {
305             moveZero(4);
306           } else {
307             moveZero(2);
308           }
309         }
310         for (int i = 0; i < moves.length; i++) {
311           moveZero(moves[i]);
312         }
313       }
314
315       if (num == 9 || num == 10) {
316         while (getRow(0) != 4) {
317           moveZero(3);
318         }
319         while (getCol(0) != (num % 7)) {
320           if (getCol(0) > (num % 7)) {
321             moveZero(4);
322           } else {
323             moveZero(2);
324           }
325         }
326         for (int i = 0; i < movesDown.length; i++) {
327           moveZero(movesDown[i]);
328         }
329       }
330     }
331
332     public void moveZero(int num) { // 1=вверх 2=вправо 3=вниз 4=влево
333       int[] temp = new int[solution.length + 1];
334       System.arraycopy(solution, 0, temp, 0, solution.length);
335       solution = temp;
336       int nullPos = getIndex(0);
337       if (num == 1) {
338         cells[nullPos] = cells[nullPos - 4];
339         cells[nullPos - 4] = 0;
340       }
341       if (num == 2) {
342         cells[nullPos] = cells[nullPos + 1];
343         cells[nullPos + 1] = 0;
344       }
345       if (num == 3) {
346         cells[nullPos] = cells[nullPos + 4];
347         cells[nullPos + 4] = 0;
348       }
349       if (num == 4) {
350         cells[nullPos] = cells[nullPos - 1];
351         cells[nullPos - 1] = 0;
352       }
```

```
353       solution[solution.length - 1] = cells[nullPos];
354       if (solution.length > 2 && solution[solution.length - 2] ==
              cells[nullPos]) {
355         temp = new int[solution.length - 2];
356         System.arraycopy(solution, 0, temp, 0, solution.length - 2);
357         solution = temp;
358       }
359     }
360
361     public int getIndex(int num) {
362       int pos = -1;
363       for (int i = 0; i < 16; i++) {
364         if (cells[i] == num) {
365           pos = i;
366         }
367       }
368       return pos;
369     }
370
371     public int getRow(int num) {
372       int pos = getIndex(num);
373       return (pos - (pos % 4)) / 4 + 1;
374     }
375
376     public int getCol(int num) {
377       int pos = getIndex(num);
378       return pos % 4 + 1;
379     }
380
381     public boolean checkRow(int num) {
382       for (int i = 1; i <= 4; i++) {
383         if (cells[(num - 1) * 4 + i] != (num - 1) * 4 + i + 1) {
384           return false;
385         }
386       }
387       return true;
388     }
389
390     public void print() {
391       for (int i = 0; i < 4; i++) {
392         for (int j = 0; j < 4; j++) {
393           System.out.print(cells[i * 4 + j] + " ");
394         }
395         System.out.print("\n");
396       }
397       System.out.print("\n");
398     }
399
400     public boolean isSolvable() {
401       int n = 0;
402       for (int i = 0; i < 16; i++) {
403         for (int j = i + 1; j < 16; j++) {
```

```
404           if (cells[j] < cells[i] && cells[i] != 0 && cells[j] != 0) {
405             n++;
406           }
407         }
408       }
409       return (n + getRow(0)) % 2 != 1;
410     }
411   }
```

4. ВЫВОД

В результате выполнения данной лабораторной работы изучены и реализованы различные алгоритмы поиска подстроки в строке, такие как алгоритм Бойера-Мура и алгоритм Кнута-Морриса-Пратта. Все методы применены к различным наборам данных. Также реализован алгоритм, решающий головоломку «Пятнашки».