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 3.

Методы поиска подстроки в строке.

Выполнил:

студент группы БВТ1902

Долматов Лев Евгеньевич

Москва
2021

Описание

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. Реализовать алгоритм решения игры «Пятнашки».

Код

```
package com.company;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        indexesOf ind = new indexesOf();
        String stroka;
        String poisk;
        Scanner scan = new Scanner(System.in);
        Lab3.main();
       /* stroka = scan.nextLine();
        poisk = scan.nextLine();
        char[] strokaa = stroka.toCharArray();
        char[] poiskk = poisk.toCharArray();
            System.out.println(ind.indexesOff(strokaa, poiskk));
            System.out.println("Дальше Мур");
            Mur.find();*/
```

```
//      int[] arr = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0};
//      int[] arr = {5, 1, 2, 3, 9, 6, 7, 4, 13, 10, 11, 8, 14, 15, 0, 12};
//      int[] arr = {7, 3, 5, 12, 6, 8, 14, 13, 2, 11, 9, 1, 0, 10, 4, 15};
        int[] arr = {1, 2, 3, 4, 5, 6, 7, 8, 13, 9, 11, 12, 10, 14, 15, 0};
//      int[] arr = {5, 1, 3, 4, 0, 2, 6, 8, 7, 10, 15, 11, 9, 13, 14, 12};
//      int[] arr = {11, 9, 4, 6, 3, 15, 7, 13, 2, 10, 0, 8, 5, 12, 1, 14}; //сложные
//      int[] arr = {5, 9, 8, 14, 0, 6, 12, 3, 13, 11, 1, 10, 15, 2, 7, 4};
//      int[] arr = {7, 1, 4, 15, 10, 12, 3, 14, 5, 6, 0, 11, 2, 13, 8, 9};
//      int[] arr = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,0};
        double inv = 0;
        for (int i = 0; i < 16; i++) {
            if (arr[i] != 0)
                for (int j = 0; j < i; ++j)
                    if (arr[j] > arr[i])
                        inv += 1;
        }
        for (int i = 0; i < 16; ++i) {
            if (arr[i] == 0)
                inv += 1 + i / 4;
        }

        int[][] arr1 = new int[4][4];
        int k = 0;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                arr1[i][j] = arr[k];
                k++;
            }
        }
        if(inv%2==0) {
```

```
            System.out.println(inv);
            new graphSearch(arr1);
        }
        else{
            System.out.println("Нет вариантов");
        }
    }
    }
```

```
package com.company;
import java.util.*;
public class Lab3 {

    public static int[] prefixFunction(String str) {
        int[] prefixFunc = new int[str.length()];
        for (int i = 1; i < str.length(); i++) {
            int j = prefixFunc[i - 1];

            while (j > 0 && str.charAt(i) != str.charAt(j)) {
                j = prefixFunc[j - 1];
            }

            if (str.charAt(i) == str.charAt(j)) {
                j += 1;
            }

            prefixFunc[i] = j;
```

```
    }

    return prefixFunc;
}

public static List<Integer> KMPSearch(String text, String pattern) {
    int[] prefixFunc = prefixFunction(pattern);
    ArrayList<Integer> occurrences = new ArrayList<Integer>();
    int j = 0;
    for (int i = 0; i < text.length(); i++) {
        while (j > 0 && text.charAt(i) != pattern.charAt(j)) {
            j = prefixFunc[j - 1];
        }
        if (text.charAt(i) == pattern.charAt(j)) {
            j += 1;
        }
        if (j == pattern.length()) {
            occurrences.add(i - j + 1);
            j = prefixFunc[j - 1];
        }
    }
    return occurrences;
}

public static int BMSearch(String T, String P)
{
    int i = P.length() -1;
    int j = P.length() -1;
    do
    {
```

```
        if (P.charAt(j) == T.charAt(i))
        {
            if (j == 0)
            {
                return i;
            }
            else
            {
                i--;
                j--;
            }
        }
        else
        {
            i = i + P.length() - min(j, 1+last(T.charAt(i), P));
            j = P.length()-1;
        }
    } while(i <= T.length()-1);

    return -1;
}
public static int last(char c, String P)
{
    for (int i=P.length()-1; i>=0; i--)
    {
        if (P.charAt(i) == c)
        {
            return i;
        }
    }
```

```
        return -1;
    }
    public static int min(int a, int b)
    {
        if (a < b)
            return a;
        else if (b < a)
            return b;
        else
            return a;
    }

    public static void main () {

        Scanner scan = new Scanner(System.in);
        System.out.println("Введите строку:");
        String str1 = scan.nextLine();
        System.out.println("Введите подстроку:");
        String str2 = scan.nextLine();

        long time1 = System.nanoTime();
        System.out.println("Поиск Кнута-Морриса-Пратта:
"+KMPSearch(str1,str2) +"\nВремя: "+(System.nanoTime()-time1) +"ns");
        long time2 = System.nanoTime();
        System.out.println("Поиск упрощенный Бойера-Мура:
"+BMSearch(str1,str2)+"\nВремя: "+(System.nanoTime()-time2) +"ns");
        long time3 = System.nanoTime();
        System.out.println("Стандартный поиск:
"+str1.indexOf(str2)+"\nВремя: "+(System.nanoTime()-time3) +"ns");
    }
```

```
}
```

```
package com.company;

import java.util.*;

class attempts {
    public int[][] array;
    public ArrayList<Integer> path = new ArrayList<>();
    //lastAct: act,
    public int opt;

    public attempts(int[][] arr, ArrayList<Integer> a, int b) {
        this.array = arr;
        this.path = a;
        //lastAct: act,
        this.opt = b;
    }
}

public class graphSearch {
    public static boolean mrt(int[][] a, int[][] b) {
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if (a[i][j] != b[i][j]) {
                    return false;
                }
```

```
        }
    }
    return true;
}

public static void Vivod(int[][] a) {
    for (int i = 0; i < 4; i++) {
        System.out.println(Arrays.toString(a[i]));
    }
}
public static int[][] Go(int[][] a) {
    int[][] b = new int[4][4];
    for (int i = 0; i < 4; i++) {
        for(int j =0;j<4;j++){
            b[i][j]=a[i][j];
        }
    }
    return b;
}
public static ArrayList<Integer> Go1(ArrayList<Integer> a) {
    ArrayList<Integer> b = new ArrayList<>();
    for (int i = 0; i <a.size(); i++) {
        b.add(a.get(i));
    }
    return b;
}
public static boolean finder(ArrayList<attempts> array, int[][] sought) {
    for (int i = 0; i < array.size(); i++) {
        if (mrt(array.get(i).array, sought)) {
            return false;
```

```
            }
        }
        return true;
    }

    public static int optimal(int[][] array) {
        int counter = 0;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                for (int l = 0; l < 4; l++) {
                    if (array[0][l] == (4 * i + j + 1)) {
                        counter += Math.abs(i) + Math.abs(j - l);
                    }
                    if (array[1][l] == (4 * i + j + 1)) {
                        counter += Math.abs(i - 1) + Math.abs(j - l);
                    }
                    if (array[2][l] == (4 * i + j + 1)) {
                        counter += Math.abs(i - 2) + Math.abs(j - l);
                    }
                    if (array[3][l] == (4 * i + j + 1)) {
                        counter += Math.abs(i - 3) + Math.abs(j - l);
                    }
                }
            }
        }

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 3; j++) {
                if (array[i][j] > array[i][j + 1] && array[i][j] != 0 && array[i][j + 1]
!= 0) {
```

```
                counter += 2;
            return counter;
        }

        public graphSearch(int[][] arr1) {
            Vivod(arr1);
            int[][] answer = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15,
0}};

            ArrayList<attempts> queue = new ArrayList<>();
            ArrayList<Integer> a = new ArrayList<>(0);
            attempts quese1 = new attempts(arr1, a, 0);
            queue.add(quese1);
            int l = 0;
            ArrayList<attempts> chekPosition = new ArrayList<>();
            //System.out.println(optimal(arr1));
            while (true) {
                attempts current;
                current = queue.remove(0);
                chekPosition.add(current);
                l++;
//              System.out.println(l);
                if (mrt(current.array, answer)) {
                    System.out.println(current.path);
                    return;
                }
                int[] indexOfZeros = {0, 0};
                for (int i = 0; i < 4; i++) {
                    for (int j = 0; j < 4; j++)
                        if (current.array[i][j] == 0) {
```

```
                    indexOfZeros[0] = i;
                    indexOfZeros[1] = j;
                    break;
                }
            }
            if (indexOfZeros[0] < 3 /*&& current.act != = 2*/) {
                int[][] newArray = Go(current.array);
                newArray[indexOfZeros[0]][indexOfZeros[1]] =
newArray[indexOfZeros[0] + 1][indexOfZeros[1]];
                newArray[indexOfZeros[0] + 1][indexOfZeros[1]] = 0;
                int action = newArray[indexOfZeros[0]][indexOfZeros[1]];
                ArrayList<Integer> newPath = Go1(current.path);
                newPath.add(action);
                if (finder(chekPosition, newArray) && finder(queue, newArray)) {
                    queue.add(new attempts(newArray, newPath,
optimal(newArray)));
                }
            }
            if (indexOfZeros[0] > 0 /*&& current.act != = 1*/) {
                int[][] newArray = Go(current.array);
                newArray[indexOfZeros[0]][indexOfZeros[1]] =
newArray[indexOfZeros[0] - 1][indexOfZeros[1]];
                newArray[indexOfZeros[0] - 1][indexOfZeros[1]] = 0;
                int action = newArray[indexOfZeros[0]][indexOfZeros[1]];
                ArrayList<Integer> newPath = Go1(current.path);
                newPath.add(action);
                if (finder(chekPosition, newArray) && finder(queue, newArray)) {
                    queue.add(new attempts(newArray, newPath,
optimal(newArray)));
                }
```

```
            }
            if (indexOfZeros[1] < 3 /*&& current.act != = 4*/) {
                int[][] newArray = Go(current.array);
                newArray[indexOfZeros[0]][indexOfZeros[1]] =
newArray[indexOfZeros[0]][indexOfZeros[1] + 1];
                newArray[indexOfZeros[0]][indexOfZeros[1] + 1] = 0;
                int action = newArray[indexOfZeros[0]][indexOfZeros[1]];
                ArrayList<Integer> newPath = Go1(current.path);
                newPath.add(action);
                if (finder(chekPosition, newArray) && finder(queue, newArray)) {
                    queue.add(new attempts(newArray, newPath,
optimal(newArray)));
                }

            }
            if (indexOfZeros[1] > 0 /*&& current.act != = 3*/) {
                int[][] newArray = Go(current.array);
                newArray[indexOfZeros[0]][indexOfZeros[1]] =
newArray[indexOfZeros[0]][indexOfZeros[1] - 1];
                newArray[indexOfZeros[0]][indexOfZeros[1] - 1] = 0;
                int action = newArray[indexOfZeros[0]][indexOfZeros[1]];
                ArrayList<Integer> newPath = Go1(current.path);
                newPath.add(action);
                if (finder(chekPosition, newArray) && finder(queue, newArray)) {
                    queue.add(new attempts(newArray, newPath,
optimal(newArray)));
                }
            }
            queue.sort(new Comparator<attempts>() {
                @Override
```

```
            public int compare(attempts o1, attempts o2) {
                return o1.opt - o2.opt;
            }
        });
    }
  }
}
```

```
package com.company;
import java.util.Arrays;
public class indexesOf {

    public static int[] indexesOff(char[] pattern, char[] text) {
        int[] pfl = pfl(pattern);
        int[] indexes = new int[text.length];
        int size = 0;
        int k = 0;
        System.out.println("eeee");
        for (int i = 0; i < text.length; ++i) {
            while (pattern[k] != text[i] && k > 0) {
                k = pfl[k - 1];
            }
            if (pattern[k] == text[i]) {
                k = k + 1;
                if (k == pattern.length) {
                    indexes[size] = i + 1 - k;
                    size += 1;
```

```
                k = pfl[k - 1];
            }
        }
        else {
            k = 0;
        }
    }
    System.out.println(Arrays.toString(indexes));
    System.out.println("сайз" + size);

    return Arrays.copyOfRange(indexes, 0, size);
}


public static int[] pfl(char[] text) {
    int[] pfl = new int[text.length];
    pfl[0] = 0;
    for (int i = 1; i < text.length; ++i) {
        int k = pfl[i - 1];
        while (text[k] != text[i] && k > 0) {
            k = pfl[k - 1];
        }
        if (text[k] == text[i]) {
            pfl[i] = k + 1;
        }
        else {
            pfl[i] = 0;
        }
    }
  for (int i = 0; i < pfl.length; i++){
```

```
        System.out.print(pfl[i]);}
        return pfl;
    }
}
```

Вывод

Выполнив данную лабораторную работу, я научился реализовывать различные методы поиска подстроки в строке двумя различными алгоритмами. Так же воспользовался алгоритмом A* для решения второго задания с игрой «Пятнашки».