Федеральное агентство связи

Государственное бюджетное образовательное учреждение высшего Образование

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «МКиИТ»

дисциплина «СиАОД»

Отчет по Лабораторной работе №3

Подготовил студент

группы БВТ1902: Капленко Е. М.

Руководитель: Мкртчян Г. М.

Москва 2020

Лабораторная работа 3. Методы поиска подстроки в строке.

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:

1. Кнута-Морриса-Пратта
2. Упрощенный Бойера-Мура

Задание 2 «Пятнашки»

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. Например, массив [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0] представляет собой «решенную» позицию элементов.

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.

В ходе работы написаны два алгоритма поиска подстроки в строке:

Кнута-Морриса-Пратта

```
Scanner sc = new Scanner(System.in);
    System.out.println("Введите строку:");
    String s = sc.nextLine();

    System.out.println("Введите подстроку:");
    String pattern = sc.nextLine();

    System.out.println("Использовать чувствительность регистра?");
    String o = sc.nextLine();
    if (o.equalsIgnoreCase("Да")) {
        String poln = pattern + "#" + s;
        int[] p = prefixFunction(poln);
        int otv1 = 0;
        for (; p[otv1] != pattern.length(); otv1++)
            if (otv1 == poln.length()-1) {
                System.out.println("Нет совпадений");
                break;
            }
        if (p[otv1] == pattern.length()) {
            otv1 = otv1 - 2 * pattern.length();
            int otv2 = s.indexOf(pattern);
            System.out.println("Индекс первого входа " + otv1 + " И в java "
+ otv2);
        }
    }else {
            String ss = s.toLowerCase();
            String sss = pattern.toLowerCase();
            String poln = sss + "#" + ss;
            int[] p = prefixFunction(poln);
            int otv1 = 0;
            for (; p[otv1] != pattern.length(); otv1++)
                if (otv1 == poln.length()-1) {
                    System.out.println("Нет совпадений");
                    break;
                }
            if (p[otv1] == pattern.length()) {
                otv1 = otv1 - 2 * pattern.length();
                int otv2 = ss.indexOf(sss);
                System.out.println("Индекс первого входа " + otv1 + " И в
java " + otv2);
            }
        }

    }
public static int[] prefixFunction(String s) {
    int[] p = new int[s.length()];
    int k = 0;
    for(int i = 1; i < s.length();i++) {
        while (k>0 && s.charAt(k) != s.charAt(i))
            k=p[k-1];
        if (s.charAt(k) == s.charAt(i))
            k++;
        p[i] = k;
    }
    return p;
}
```

Бойера-Мура

```
System.out.println("Введите строку:");
    String source = sc.nextLine();

    System.out.println("Введите подстроку:");
    String template = sc.nextLine();

    int[] p = function(template);

    HashMap<Character, Integer> table = new HashMap<>();
    for (int i = 0; i< template.length(); i++){
        if (table.containsKey(template.charAt(i)))
            continue;
        table.put(template.charAt(i),p[i]);
    }

    int k = ishetOtvet(template,source,table);

    if (k>=0)    System.out.print("Вот индекс " + k);
    else System.out.print("Такой подстроки здесь нет");

}
public static int ishetOtvet (String template, String source,
HashMap<Character, Integer> table) {
    int j = template.length() - 1;
    while (j<source.length()) {
        for (int i = template.length() - 1; i>=0; i--, j--) {
            if (template.charAt(i) != source.charAt(j)) {
                if (table.containsKey(source.charAt(j))) {
                    j += table.get(source.charAt(j));
                } else {
                    j += template.length();
                }
                break;
            }
            if (i == 0) return j;
        }
    }
    return -1;
}
public static int[] function (String template) {
    int[] d = new int[template.length()];
    int k = 0;
    int j=0;
    int l = template.length() - 1;
    for (int i = template.length(); i > 0; i--) {
        d[i - 1] = k;
        while (j < k) {
            if (template.charAt(i - 1) == template.charAt(l - j)) {
                d[i - 1] = d[l - j];
                break;
            }
            j++;
        }
        j = 0;
        k++;
    }
    return d;
}
```

И реализовано решение пятнашек, при помощи алгоритма A*, и вспомогательных классов, таких как FifteenRules, FifteenPules2, FifteenState.

```
parseArgs(args);

    if (isReadFromStream) {
        try {
            startField = readStartState();
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
        if (sideSize == 4 && !FifteenState.checkState(startField)) {
            System.out
                    .println("\nДанное состояние нельзя привести к
терминальному.\n");
            System.exit(1);
        }
    }

    int size = sideSize * sideSize;
    terminateField = getTerminalState(sideSize, size);

    FifteenRules rules = new FifteenRules2(sideSize, terminateField);
    FifteenState startState = new FifteenState(null, sideSize);

    if (startField == null) {
        startField = generateStartState(rules, stepCount);
    }
    startState.setField(startField);

    Astar<FifteenState, FifteenRules> astar = new Astar<FifteenState,
FifteenRules>(
            rules);
    long time = System.currentTimeMillis();
    Collection<State> res = astar.search(startState);
    time = System.currentTimeMillis() - time;

    if (res == null) {
        System.out.println("Solution not found.");
        return;
    } else {
        for (State s : res) {
            System.out.println(s.toString());
        }
    }
    if (isShowStatistic) {
        System.out.println("Time: " + time + "ms");
        /* Начальное состояние за ход не считается */
        System.out.println("Solution length: " + (res.size() - 1));
        System.out
                .println("Opened states: " + astar.getClosedStatesCount());
    }
}
```

Вывод: В данной лабораторной работе я самостоятельно реализовала алгоритмы Кнута-Морриса-Пратта и Бойера-Мура, и решение игры

“Пятнашки”. Ознакомилась с понятием префиксная функция, более детально ознакомилась с A* алгоритмом.