Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Бюджетное образовательное учреждение высшего образования
Московский технический университет связи и информатики

Кафедра «Математической кибернетики и информационных технологий»

Дисциплина «Структуры и алгоритмы обработки данных»

Отчёт по лабораторной работе №3

«Методы поиска подстроки в строке»

Выполнил студент
Группы БФИ1901
Курбатов А.О.

Проверил
Кутейников И. А.

Москва 2021

Задание

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.

Например, для начального расположения элементов [1, 2, 3, 4, 5, 6, 7, 8, 13, 9, 11, 12, 10, 14, 15, 0] одним из возможных решений будет [15, 14, 10, 13, 9, 10, 14, 15] (последовательность шагов здесь: двигаем 15, двигаем 14, двигаем 10, …, двигаем 15).

Листинг

```
public class Lab3 {
    public static int kmp(String text, String find) {
        if (find.length() > text.length()) {
            return -1;
        }
        text = find + "#" + text;
        int count = 0;
        int i = 0;
        int j = -1;
        int[] pref = new int[text.length() + 1];
        pref[0] = -1;
        while (i < text.length()) {
            while (j >= 0 && text.charAt(j) != text.charAt(i)) {
                j = pref[j];
            }
            j++;
            i++;
            pref[i] = j;
        }
        for(int k = 1; k < pref.length; k++){
            if(pref[k] == find.length()) count++;
        }
        return count;
    }

    public static int bm(String text, String find) {
        if (find.length() > text.length()) {
            return -1;
        }
        int d[] = new int [256];
        for (int i = 0; i <= 255; i++) {
            d[i] = find.length();
        }
        for (int i = 0; i < find.length() - 1; i++) {
            d[find.charAt(i)] = find.length() - i - 1;
        }
        int i = find.length() - 1;
        int j = i;
        int k = i;
        while (j >= 0 && i <= text.length() - 1) {
            j = find.length() - 1;
            k = i;
            while (j >= 0 && text.charAt(k) == find.charAt(j)) {
                k--;
                j--;
            }
            i += d[text.charAt(i)];
        }
        if (k >= text.length() - find.length()) {
            return -1;
        } else {
            return k + 1;
        }
    }

    public static void main(String[] args) {
        String text1 = "aabaabaaaabaabaaab";
```

```
        String find1 = "aabaa";
        String text2 = "Hoola-Hoola girl likes Hooligans";
        String find2 = "Hooligan";

        System.out.println("Алгоритм Кнута-Морриса-Пратта:");
        System.out.println("Количество " + find1 + " в строке " + text1 + ": \n"
+ kmp(text1, find1));
        System.out.println("\nУпрощенный алгоритм Бойера-Мура:");
        System.out.println("Индекс, на котором находится  " + find2 + " в строке
" + text2 + ": \n" + bm(text2, find2));

        int[][] blocks = new int[][]{{1, 2, 3, 4}, {5, 6, 7, 8}, {13, 9, 11,
12}, {10, 14, 15, 0}};
        Board initial = new Board(blocks);
        Solver solver = new Solver(initial);

        System.out.println();
        for (Board board : solver.solution())
            System.out.println(board);
    }
}

import java.util.*;

public class Solver {

    private Board initial;
    private List<Board> result = new ArrayList<Board>();
    private class ITEM{
        private ITEM prevBoard;
        private Board board;

        private ITEM(ITEM prevBoard, Board board) {
            this.prevBoard = prevBoard;
            this.board = board;
        }

        public Board getBoard() {
            return board;
        }


    }

    public Solver(Board initial) {
        this.initial = initial;

        if(!isSolvable()) return;

        PriorityQueue<ITEM> priorityQueue = new PriorityQueue<ITEM>(10, new
Comparator<ITEM>() {
            @Override
            public int compare(ITEM o1, ITEM o2) {
                return new Integer(measure(o1)).compareTo(new
Integer(measure(o2)));
            }
        });

        priorityQueue.add(new ITEM(null, initial));
```

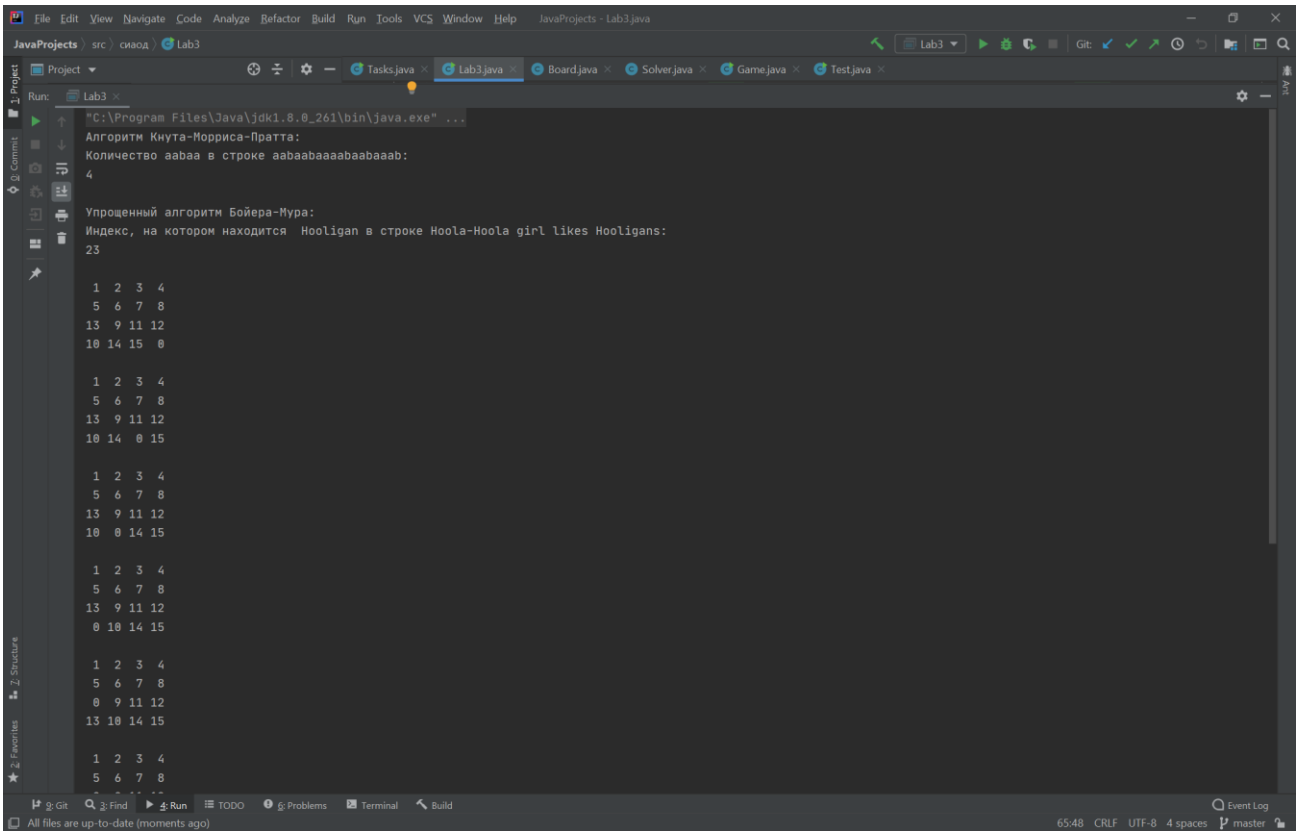
```
        while (true){
            ITEM board = priorityQueue.poll();

            if(board.board.isGoal()) {
                itemToList(new ITEM(board, board.board));

                return;
            }

            Iterator iterator = board.board.neighbors().iterator();
            while (iterator.hasNext()){
                Board board1 = (Board) iterator.next();
                if(board1!= null && !containsInPath(board, board1))
                    priorityQueue.add(new ITEM(board, board1));
            }

        }
    }

    private static int measure(ITEM item){
        ITEM item2 = item;
        int c= 0;
        int measure = item.getBoard().h();
        while (true){
            c++;
            item2 = item2.prevBoard;
            if(item2 == null) {
                return measure + c;
            }
        }
    }
    private void itemToList(ITEM item){
        ITEM item2 = item;
        while (true){
            item2 = item2.prevBoard;
            if(item2 == null) {
                Collections.reverse(result);
                return;
            }

            result.add(item2.board);
        }
    }
        private boolean containsInPath(ITEM item, Board board){
        ITEM item2 = item;
        while (true){
            if(item2.board.equals(board)) return true;
            item2 = item2.prevBoard;
            if(item2 == null) return false;
        }
    }


    public boolean isSolvable() {
        return true;
    }

    public int moves() {
```

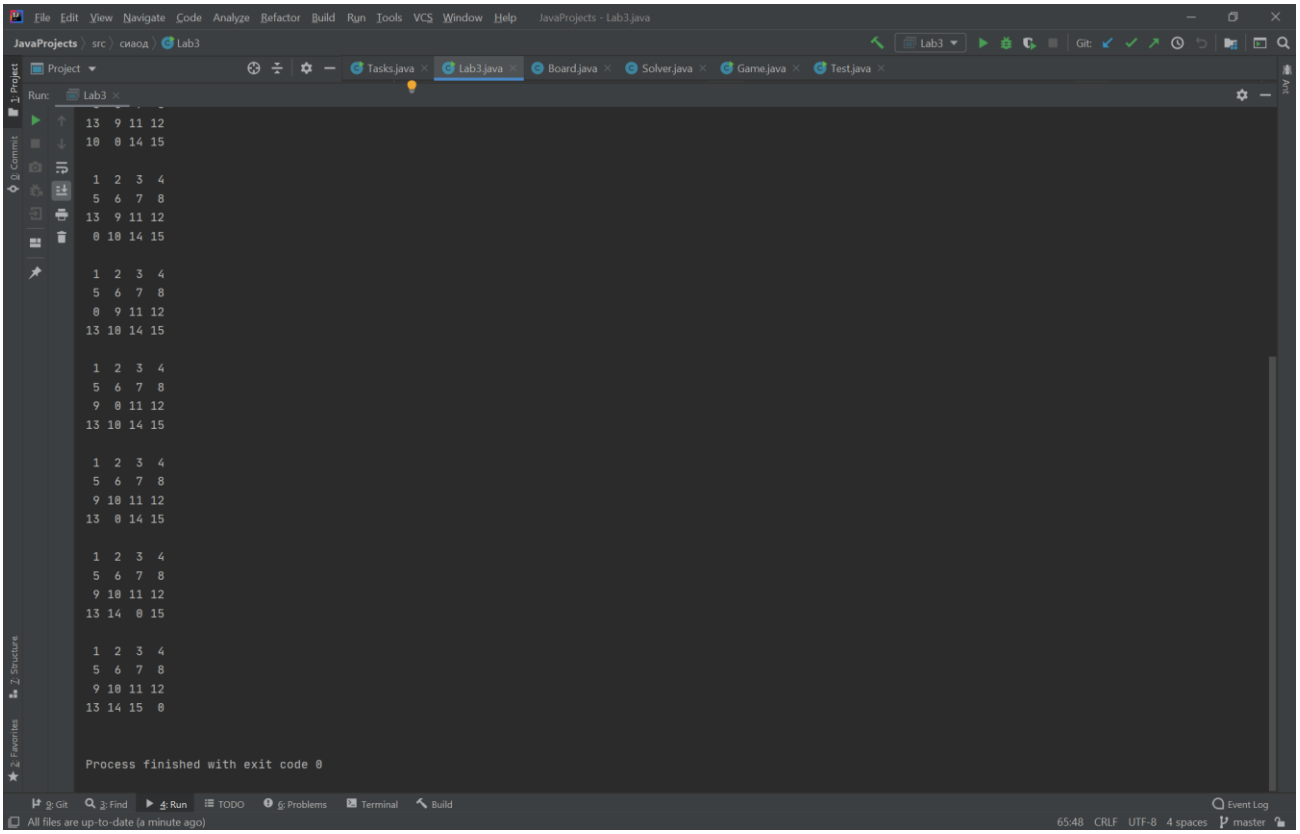
```
        if(!isSolvable()) return -1;
        return result.size() - 1;
    }


    public Iterable<Board> solution() {
        return result;
    }
}

import java.util.HashSet;
import java.util.Set;

public class Board {
    private int[][] blocks;
    private int zeroX;
    private int zeroY;
    private int h;

    public Board(int[][] blocks) {
        int[][] blocks2 = deepCopy(blocks);
        this.blocks = blocks2;

        h = 0;
        for (int i = 0; i < blocks.length; i++) {
            for (int j = 0; j < blocks[i].length; j++) {
                if (blocks[i][j] != (i*dimension() + j + 1) && blocks[i][j] !=
0) {
                    h += 1;
                }
                if (blocks[i][j] == 0) {
                    zeroX = (int) i;
                    zeroY = (int) j;
                }
            }
        }
    }


    public int dimension() {
        return blocks.length;
    }

    public int h() {
        return h;
    }

    public boolean isGoal() {
        return h == 0;
    }


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Board board = (Board) o;
```

```
        if (board.dimension() != dimension()) return false;
        for (int i = 0; i < blocks.length; i++) {
            for (int j = 0; j < blocks[i].length; j++) {
                if (blocks[i][j] != board.blocks[i][j]) {
                    return false;
                }
            }
        }

        return true;
    }

    public Iterable<Board> neighbors() {
        Set<Board> boardList = new HashSet<Board>();
        boardList.add(chng(getNewBlock(), zeroX, zeroY, zeroX, zeroY + 1));
        boardList.add(chng(getNewBlock(), zeroX, zeroY, zeroX, zeroY - 1));
        boardList.add(chng(getNewBlock(), zeroX, zeroY, zeroX - 1, zeroY));
        boardList.add(chng(getNewBlock(), zeroX, zeroY, zeroX + 1, zeroY));

        return boardList;
    }

    private int[][] getNewBlock() {
        return deepCopy(blocks);
    }

    private Board chng(int[][] blocks2, int x1, int y1, int x2, int y2) {
        if (x2 > -1 && x2 < dimension() && y2 > -1 && y2 < dimension()) {
            int t = blocks2[x2][y2];
            blocks2[x2][y2] = blocks2[x1][y1];
            blocks2[x1][y1] = t;
            return new Board(blocks2);
        } else
            return null;

    }


    public String toString() {
        StringBuilder s = new StringBuilder();
        for (int i = 0; i < blocks.length; i++) {
            for (int j = 0; j < blocks.length; j++) {
                s.append(String.format("%2d ", blocks[i][j]));
            }
            s.append("\n");
        }
        return s.toString();
    }

    private static int[][] deepCopy(int[][] original) {
        if (original == null) {
            return null;
        }

        final int[][] result = new int[original.length][];
        for (int i = 0; i < original.length; i++) {
            result[i] = new int[original[i].length];
            for (int j = 0; j < original[i].length; j++) {
                result[i][j] = original[i][j];
```

```
                }
            }
            return result;
        }
}
```

Результат выполнения работы





Вывод

В ходе данной лабораторной работы мы изучили методы поиска подстроки в строке: Кнута-Морриса-Пратта и упрощенный Бойера-Мура. Мы реализовали указанные алгоритмы на языке программирования java, а также реализовали нахождение решения для игры «Пятнашки».