Отчёт по лабораторной работе №4

Выполнил: Трусов Михаил Павлович, ПИН-22

Задание 1. Составить программу решения задачи коммивояжера для графа, заданного матрицей смежности

	1	2	3	4	5
1	∞	7	12	25	10
2	10	∞	9	5	11
3	13	8	∞	6	4
4	6	11	15	∞	15
5	5	9	12	17	∞

Результат выполнения программы:

```
Кратчайший путь: [2, 4, 3, 5, 1, 2], длина = 36
```

Программный код:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

class OptimalPathFinder {

    private final Map<String, Map<String, Integer>> distances;

    public OptimalPathFinder(Map<String, Map<String, Integer>> distances) {
        this.distances = distances;
    }

    private static <T> void swap(T[] array, int first, int second) {
        T temp = array[first];
        array[first] = array[second];
        array[second] = temp;
    }

    private static <T> void allPermutationsHelper(T[] permutation, List<T[]>
permutations, int n) {
        if (n <= 0) {
```

```
            permutations.add(permutation);
            return;
        }
        T[] tempPermutation = Arrays.copyOf(permutation, permutation.length);
        for (int i = 0; i < n; i++) {
            swap(tempPermutation, i, n - 1);
            allPermutationsHelper(tempPermutation, permutations, n - 1);
            swap(tempPermutation, i, n - 1);
        }
    }

    private static <T> List<T[]> permutations(T[] original) {
        List<T[]> permutations = new ArrayList<>();
        allPermutationsHelper(original, permutations, original.length);
        return permutations;
    }

    public int pathDistance(String[] path) {
        String last = path[0];
        int distance = 0;
        for (String next : Arrays.copyOfRange(path, 1, path.length)) {
            distance += distances.get(last).get(next);
            last = next;
        }
        return distance;
    }

    public String[] findShortestPath() {
        String[] cities = distances.keySet().toArray(String[]::new);

        List<String[]> paths = permutations(cities);

        String[] shortestPath = null;
        int minDistance = Integer.MAX_VALUE;

        for (String[] path : paths) {
            int distance = pathDistance(path);

            distance += distances.get(path[path.length - 1]).get(path[0]);

            if (distance < minDistance) {
                minDistance = distance;
                shortestPath = path;
            }
        }

        shortestPath = Arrays.copyOf(shortestPath, shortestPath.length + 1);
        shortestPath[shortestPath.length - 1] = shortestPath[0];

        return shortestPath;
    }

    public static void main(String[] args) {
        Map<String, Map<String, Integer>> vtDistances = new HashMap<>();
        vtDistances.put("1", Map.of(
                "2", 7,
                "3", 12,
                "4", 25,
                "5", 10));
        vtDistances.put("2", Map.of(
                "1", 10,
                "3", 9,
                "4", 5,
                "5", 11));
```

```
        vtDistances.put("3", Map.of(
                "1", 13,
                "2", 8,
                "4", 6,
                "5", 4));
        vtDistances.put("4", Map.of(
                "1", 6,
                "2", 11,
                "3", 15,
                "5", 15));
        vtDistances.put("5", Map.of(
                "1", 5,
                "2", 9,
                "3", 12,
                "4", 17
        ));

        OptimalPathFinder tsp = new OptimalPathFinder(vtDistances);

        String[] shortestPath = tsp.findShortestPath();

        int distance = tsp.pathDistance(shortestPath);
        System.out.println("Кратчайший путь: " +
Arrays.toString(shortestPath) + ", длина = " + distance);
    }
}
```

Задание 2. Решить задачу коммивояжера для задач, представленных в следующем разделе (вариант 13 - граф задан матрицей смежности размером 5×5, в которой помимо диагональных элементов удалены еще два ребра)

Результат выполнения программы:

```
Кратчайший путь: [2, 3, 4, 5, 1, 2], длина = 18
```

Программный код:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

class OptimalPathFinder {

    private final Map<String, Map<String, Integer>> distances;

    public OptimalPathFinder(Map<String, Map<String, Integer>> distances) {
        this.distances = distances;
    }

    private static <T> void swap(T[] array, int first, int second) {
        T temp = array[first];
        array[first] = array[second];
        array[second] = temp;
    }

    private static <T> void allPermutationsHelper(T[] permutation, List<T[]>
```

```
permutations, int n) {
        if (n <= 0) {
            permutations.add(permutation);
            return;
        }
        T[] tempPermutation = Arrays.copyOf(permutation, permutation.length);
        for (int i = 0; i < n; i++) {
            swap(tempPermutation, i, n - 1);
            allPermutationsHelper(tempPermutation, permutations, n - 1);
            swap(tempPermutation, i, n - 1);
        }
    }

    private static <T> List<T[]> permutations(T[] original) {
        List<T[]> permutations = new ArrayList<>();
        allPermutationsHelper(original, permutations, original.length);
        return permutations;
    }

    public int pathDistance(String[] path) {
        String last = path[0];
        int distance = 0;
        for (String next : Arrays.copyOfRange(path, 1, path.length)) {
            distance += distances.get(last).get(next);
            last = next;
        }
        return distance;
    }

    public String[] findShortestPath() {
        String[] cities = distances.keySet().toArray(String[]::new);

        List<String[]> paths = permutations(cities);

        String[] shortestPath = null;
        int minDistance = Integer.MAX_VALUE;

        for (String[] path : paths) {
            int distance = pathDistance(path);

            distance += distances.get(path[path.length - 1]).get(path[0]);

            if (distance < minDistance) {
                minDistance = distance;
                shortestPath = path;
            }
        }

        shortestPath = Arrays.copyOf(shortestPath, shortestPath.length + 1);
        shortestPath[shortestPath.length - 1] = shortestPath[0];

        return shortestPath;
    }

    public static void main(String[] args) {
        Map<String, Map<String, Integer>> vtDistances = new HashMap<>();
        vtDistances.put("1", Map.of(
                "2", 2,
                "3", 14,
                "4", 6,
                "5", 999999999));
        vtDistances.put("2", Map.of(
                "1", 1,
                "3", 2,
```

```
                "4", 7,
                "5", 6));
        vtDistances.put("3", Map.of(
                "1", 16,
                "2", 999999999,
                "4", 7,
                "5", 2));
        vtDistances.put("4", Map.of(
                "1", 6,
                "2", 13,
                "3", 8,
                "5", 4));
        vtDistances.put("5", Map.of(
                "1", 3,
                "2", 7,
                "3", 13,
                "4", 6
        ));

        OptimalPathFinder tsp = new OptimalPathFinder(vtDistances);

        String[] shortestPath = tsp.findShortestPath();

        int distance = tsp.pathDistance(shortestPath);
        System.out.println("Кратчайший путь: " +
Arrays.toString(shortestPath) + ", длина = " + distance);
    }
}
```

Задание 3. Составить функциональную таблицу и функциональную диаграмму, программу (вариант 13 - задан произвольный двоичный код. Получить на ленте зеркальное отображение кода)

Функциональная таблица:

Слово: 101100 | Поместить | Очистить ленту

Алфавит: 01 | Задать

Начальное состояние: q0

Машина: Запустить | Сделать шаг | Сброс | Сохранить | Загрузить

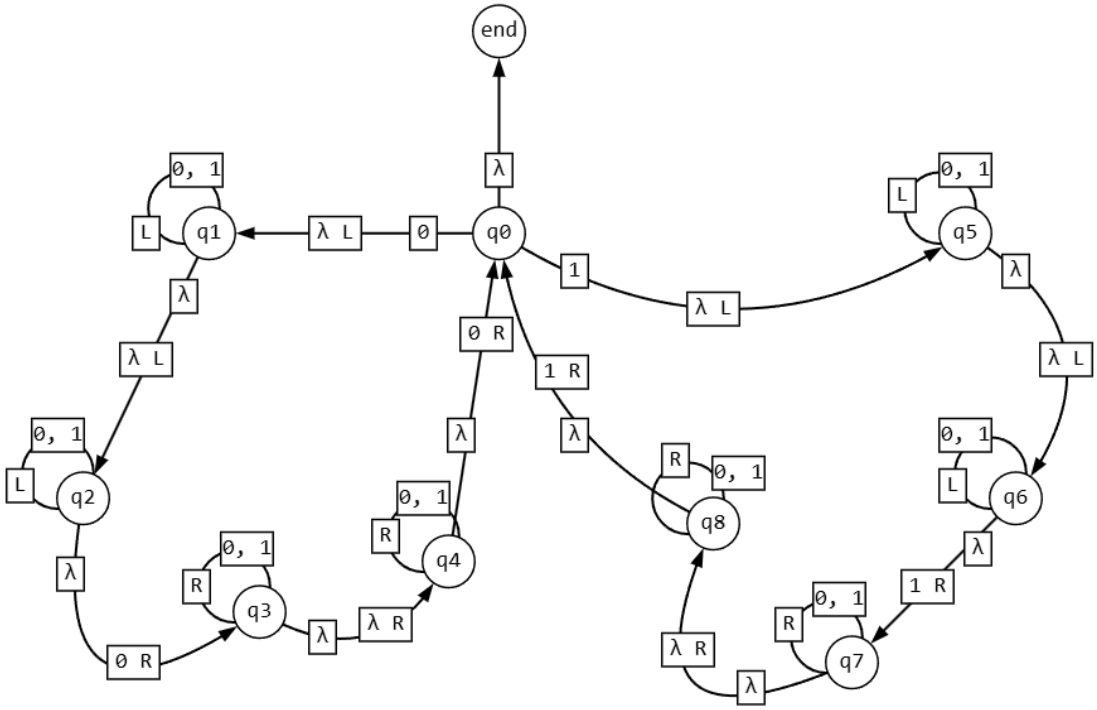
Состояние ⊕	0	1	λ
q0 ⊖	λ L q1	λ L q5	!
q1 ⊖	0 L q1	1 L q1	λ L q2
q2 ⊖	0 L q2	1 L q2	0 R q3
q3 ⊖	0 R q3	1 R q3	λ R q4
q4 ⊖	0 R q4	1 R q4	0 R q0
q5 ⊖	0 L q5	1 L q5	λ L q6
q6 ⊖	0 L q6	1 L q6	1 R q7
q7 ⊖	0 R q7	1 R q7	λ R q8
q8 ⊖	0 R q8	1 R q8	1 R q0

Результат работы машины Тьюринга:

λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	1	0	1	1	0	0	λ	λ	λ	λ	λ

λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	0	0	1	1	0	1	λ	1	0	1	1	0	0	λ	λ	λ	λ	λ

Функциональная диаграмма:



Результат выполнения программы:

```
101100
ΛΛΛΛΛΛΛΛΛΛΛΛΛΛ101100ΛΛΛΛΛΛΛΛΛΛΛΛΛΛ
ΛΛΛΛΛΛΛΛ001101Λ101100ΛΛΛΛΛΛΛΛΛΛΛΛΛΛ
```

Программный код:

```
import java.util.Scanner;

public class Main {

    public static String replaceAt(String line, int index, String
what_to_place) {
        char[] myNameChars = line.toCharArray();
        myNameChars[index] = what_to_place.charAt(0);
        return String.valueOf(myNameChars);
    }

    public static String q0(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0" -> q1(replaceAt(line, pointer, "λ"), pointer - 1);
            case "1" -> q5(replaceAt(line, pointer, "λ"), pointer - 1);
            case "λ" -> line;
            default -> null;
        };
    }

    public static String q1(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
```

```
            case "0", "1" -> q1(line, pointer - 1);
            case "λ" -> q2(replaceAt(line, pointer, "λ"), pointer - 1);
            default -> null;
        };
    }

    public static String q2(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q2(line, pointer - 1);
            case "λ" -> q3(replaceAt(line, pointer, "0"), pointer + 1);
            default -> null;
        };
    }

    public static String q3(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q3(line, pointer + 1);
            case "λ" -> q4(replaceAt(line, pointer, "λ"), pointer + 1);
            default -> null;
        };
    }

    public static String q4(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q4(line, pointer + 1);
            case "λ" -> q0(replaceAt(line, pointer, "0"), pointer + 1);
            default -> null;
        };
    }


    public static String q5(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q5(line, pointer - 1);
            case "λ" -> q6(replaceAt(line, pointer, "λ"), pointer - 1);
            default -> null;
        };
    }

    public static String q6(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q6(line, pointer - 1);
            case "λ" -> q7(replaceAt(line, pointer, "1"), pointer + 1);
            default -> null;
        };
    }

    public static String q7(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q7(line, pointer + 1);
            case "λ" -> q8(replaceAt(line, pointer, "λ"), pointer + 1);
            default -> null;
        };
    }

    public static String q8(String line, int pointer) {
        return switch (String.valueOf(line.charAt(pointer))) {
            case "0", "1" -> q8(line, pointer + 1);
            case "λ" -> q0(replaceAt(line, pointer, "1"), pointer + 1);
            default -> null;
        };
    }
```

```
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String num = in.nextLine();
        StringBuilder line = new StringBuilder();
        line.append("λ".repeat(15));
        line.append(num);
        line.append("λ".repeat(15));
        System.out.println(line);
        System.out.println(q0(line.toString(), 15));
    }
}
```