Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Домашняя работа № 4

По Дискретной Математике

Гамильтонов Цикл

Вариант № 20

Выполнил:

Карташев Владимир P3131

Преподаватель:

Поляков Владимир Иванович

г. Санкт-Петербург, 2023 г.

Исходная таблица соединений R:

V/V	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12
e1	0		3	3		5	1		1			
e2		0				2	1					
e3	3		0				5	2	4			3
e4	3			0	1		3					
e5				1	0	4				3		
e6	5	2			4	0	2	3	2			
e7	1	1	5	3		2	0		2		1	
e8			2			3		0	4	5		
e9	1		4			2	2	4	0	5	4	
e10					3			5	5	0		1
e11							1		4		0	
e12			3							1		0

Нахождение гамильтонова цикла

- Для начала, возьмем в **S** вершину e_1. Следовательно, в начале алгоритма $S = \{e_1^+\}$.

Напишем программу для нахождения гамильтонова цикла:

- Программа написана на языке **Java** в стиле ООП
- Программа состоит из двух классов: **Main.class** и **HamiltonianPath.class**

Main.class:

```java
/**
 * This program is used to find Hamiltonian Path and Hamiltonian Cycle
 *
 * @author Vladimir Kartashev
 * @version 1.0
 * @since 04-06-2023
 */
public class Main {
    public static void main(String[] args) {
        /**
         * The INT graphMatrix massive graph is used to contain the graph
         *
         * @param graphMatrix
         */
        int[][] graphMatrix = new int[][]{
//              Variant: 20
//              1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12
                {0, 0, 3, 3, 0, 5, 1, 0, 1, 0, 0, 0}, // 1
                {0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0}, // 2
                {3, 0, 0, 0, 0, 0, 5, 2, 4, 0, 0, 3}, // 3
                {3, 0, 0, 0, 1, 0, 3, 0, 0, 0, 0, 0}, // 4
                {0, 0, 0, 1, 0, 4, 0, 0, 0, 3, 0, 0}, // 5
                {5, 2, 0, 0, 4, 0, 2, 3, 2, 0, 0, 0}, // 6
                {1, 1, 5, 3, 0, 2, 0, 0, 2, 0, 1, 0}, // 7
                {0, 0, 2, 0, 0, 3, 0, 0, 4, 5, 0, 0}, // 8
                {1, 0, 4, 0, 0, 2, 2, 4, 0, 5, 4, 0}, // 9
                {0, 0, 0, 0, 3, 0, 0, 5, 5, 0, 0, 1}, // 10
                {0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 0, 0}, // 11
                {0, 0, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0}  // 12
        };

        /**
         * The STRING graphVerticesNames massive vertices is used to contain graph vertex names
         *
         * @param graphVerticesNames
         */
        String[] graphVerticesNames = {"e1", "e2", "e3", "e4", "e5", "e6", "e7", "e8", "e9", "e10", "e11", "e12"};


        HamiltonianPath hamiltonianPath = new HamiltonianPath();
        hamiltonianPath.hamiltonianPath(graphMatrix, graphVerticesNames);

        System.out.println();

        HamiltonianCycle hamiltonianCycle = new HamiltonianCycle();
        hamiltonianCycle.hamiltonianCycle(graphMatrix);
    }
}
```

HamiltonianPath.class:

```java
/**
 * The HamiltonianPath class is used to find Hamiltonian path
 *
 * @author Vladimir Kartashev
 * @version 1.0
 * @since 04-06-2023
 */
public class HamiltonianPath {
    /**
     * The numberOfVertices field is used to contain number of vertices
     * @param numberOfVertices
     */
    private int numberOfVertices;


    /**
     * The path massive is used to contain current path
     * @param path
     */
    private int[] path;


    /**
     * The graph massive is used to contain matrix of graph
     * @param graphMatrix
     */
    private int[][] graphMatrix;


    /**
     * The visitedVertices massive is used to contain visited vertices information
     * @param visitedVertices
     */
    private boolean[] visitedVertices;


    /**
     * The graphVerticesNames massive is used to contain graph vertex names
     * @param graphVerticesNames
     */
    private String[] graphVerticesNames;



```

...

```java
    /**
     * Main hamiltonianPath() method
     * @param graphMatrix
     * @param graphVerticesNames
     */
    public void hamiltonianPath(int[][] graphMatrix, String[] graphVerticesNames) {
        numberOfVertices = graphMatrix.length;
        path = new int[numberOfVertices];
        visitedVertices = new boolean[numberOfVertices];
        this.graphVerticesNames = graphVerticesNames;
        this.graphMatrix = graphMatrix;

        for (int i = 0; i < numberOfVertices; i++) {
            visitedVertices[i] = false;
            path[i] = -1;
        }

        path[0] = 0;
        visitedVertices[0] = true;

        if (findPath(1)) {
            printPath();
        } else {
            System.out.println("No Path Found");
        }
    }



    /**
     * Helper findPath() method
     *
     * @param vertexIndex
     * @return BOOLEAN value about whether the path is found or not
     */
    private boolean findPath(int vertexIndex) {
        if (vertexIndex == numberOfVertices) {
            return graphMatrix[path[vertexIndex - 1]][path[0]] == 1;
        }

        for (int i = 0; i < numberOfVertices; i++) {
            if (isSafe(vertexIndex, i)) {
                path[vertexIndex] = i;
                visitedVertices[i] = true;

                if (findPath(vertexIndex + 1)) {
                    return true;
                }

                visitedVertices[i] = false;
            }
        }
        return false;
    }
```

. . .

```
103         /**
104          * Helper isSafe() method
105          *
106          * Method isSafe() check that vertexIndex can be added to the path
107          *
108          * @param vertexIndex
109          * @param vertex
110          * @return BOOLEAN value that vertexIndex can be added to the path
111          */
112         private boolean isSafe(int vertexIndex, int vertex) {
113             if (graphMatrix[path[vertexIndex - 1]][vertex] == 0) {
114                 return false;
115             }
116 
117             for (int i = 0; i < vertexIndex; i++) {
118                 if (path[i] == vertex) {
119                     return false;
120                 }
121             }
122 
123             return true;
124         }
125 
126 
127 
128         /**
129          * Helper printPath() method
130          *
131          * Method printPath() print the found Hamiltonian Path
132          */
133         private void printPath() {
134             System.out.println("Hamiltonian Path: ");
135 
136             for (int i = 0; i < path.length; i++) {
137                 System.out.print(graphVerticesNames[path[i]] + " ");
138             }
139 
140             System.out.println("| " + graphVerticesNames[path[0]] + " ...");
141         }
142     }
```

Получившийся гамильтонов путь:

$S = \{e_1, e_3, e_{12}, e_{10}, e_8, e_9, e_{11}, e_7, e_2, e_6, e_5, e_4\}$

Перенумеруем вершины графа согласно полученному гамильтонову циклу таким образом, чтобы ребра гамильтонова цикла были внешними:

До	e_1	e_3	e_{12}	e_{10}	e_8	e_9	e_{11}	e_7	e_2	e_6	e_5	e_4
После	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}

Таблица соединений R с перенумерованными вершинами:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
x_1	0	1	0	0	0	1	0	1	0	1	0	1
x_2	1	0	1	0	1	1	0	1	0	0	0	0
x_3	0	1	0	1	0	0	0	0	0	0	0	0
x_4	0	0	1	0	1	1	0	0	0	0	1	0
x_5	0	1	0	1	0	1	0	0	0	1	0	0
x_6	1	1	0	1	1	0	1	1	0	1	0	0
x_7	0	0	0	0	0	1	0	1	0	0	0	0
x_8	1	1	0	0	0	1	1	0	1	1	0	1
x_9	0	0	0	0	0	0	0	1	0	1	0	0
x_{10}	1	0	0	0	1	1	0	1	1	0	1	0
x_{11}	0	0	0	1	0	0	0	0	0	1	0	1
x_{12}	1	0	0	0	0	0	0	1	0	0	1	0

Построение графа пересечений G’

1. Определим p_{2-8}, для чего в матрице R выделим подматрицу R_{2-8}.
   Ребро (x_2x_8) пересекается с (x_1x_6).
2. Определим p_{4-11}, для чего в матрице R выделим подматрицу R_{4-11}.
   Ребро (x_4x_{11}) пересекается с (x_1x_6), (x_1x_8), (x_1x_{10}), (x_2x_5), (x_2x_6), (x_2x_8).
3. Определим p_{4-6}, для чего в матрице R выделим подматрицу R_{4-6}.
   Ребро (x_4x_6) пересекается с (x_2x_5).
4. Определим p_{5-10}, для чего в матрице R выделим подматрицу R_{5-10}.
   Ребро (x_5x_{10}) пересекается с (x_1x_6), (x_1x_8), (x_2x_6), (x_2x_8), (x_4x_6).
5. Определим p_{6-10}, для чего в матрице R выделим подматрицу R_{6-10}.
   Ребро (x_6x_{10}) пересекается с (x_1x_8), (x_2x_8).
6. Определим p_{8-12}, для чего в матрице R выделим подматрицу R_{8-12}.
   Ребро (x_8x_{12}) пересекается с (x_1x_{10}), (x_4x_{11}), (x_5x_{10}), (x_6x_{10})

. . .

Найдено 11 пересечений графа:

	p_{1-6}	p_{2-8}	p_{4-11}	p_{1-8}	p_{1-10}	p_{2-5}	p_{2-6}	p_{4-6}	p_{5-10}	p_{6-10}	p_{8-12}
p_{1-6}	1	1	1	0	0	0	0	0	1	0	0
p_{2-8}	1	1	1	0	0	0	0	0	1	1	0
p_{4-11}	1	1	1	1	1	1	1	0	0	0	1
p_{1-8}	0	0	1	1	0	0	0	0	1	1	0
p_{1-10}	0	0	1	0	1	0	0	0	0	0	1
p_{2-5}	0	0	1	0	0	1	0	1	0	0	0
p_{2-6}	0	0	1	0	0	0	1	0	1	0	0
p_{4-6}	0	0	0	0	0	1	0	1	1	0	0
p_{5-10}	1	1	0	1	0	0	1	1	1	0	1
p_{6-10}	0	1	0	1	0	0	0	0	0	1	1
p_{8-12}	0	0	1	0	1	0	0	0	1	1	1

Построение семейства ψG

1. В 1 строке ищем первый нулевой элемент - r_{1-4}.
   Записываем дизъюнкцию $M_{1-4} = r_1 \vee r_4 =$
   $= 11100000100 \vee 00110000110 = 11110000110$
2. В строке M_{1-4} находим номера нулевых элементов, составляем список $J' = \{5, 6, 7, 8, 11\}$.
   Записываем дизъюнкцию $M_{1-4-5} = M_{1-4} \vee r_5 =$
   $= 11110000110 \vee 00101000001 = 11111000111$
3. В строке M_{1-4-5} находим номера нулевых элементов, составляем список $J' = \{6, 7, 8\}$.
   Записываем дизъюнкцию $M_{1-4-5-6} = M_{1-4-5} \vee r_6 =$
   $= 11111000111 \vee 00100101000 = 11111101111$

4. В строке $M_{1-4-5-6}$ находим номера нулевых элементов, составляем список $J' = \{7\}$.
   Записываем дизъюнкцию $M_{1-4-5-6-7} = M_{1-4-5-6} \vee r_7 =$
   $= 11111101111 \vee 00100010100 = 11111111111$
5. В строке $M_{1-4-5-6-7}$ все регистры равны 1. Построено
   $\psi_1 = \{u_{1-6}, u_{1-8}, u_{1-10}, u_{2-5}, u_{2-6}\}$

Делаем также начиная с ψ_2 до ψ_{13}. Тогда, семейство максимальных внутренне устойчивых множеств ψ_G будет:

$\psi_1 = \{u_{1-6}, u_{1-8}, u_{1-10}, u_{2-5}, u_{2-6}\}$
$\psi_2 = \{u_{1-6}, u_{1-8}, u_{1-10}, u_{2-6}, u_{4-6}\}$
$\psi_3 = \{u_{1-6}, u_{1-8}, u_{2-5}, u_{2-6}, u_{8-12}\}$
$\psi_4 = \{u_{1-6}, u_{1-8}, u_{2-6}, u_{4-6}, u_{8-12}\}$
$\psi_5 = \{u_{1-6}, u_{1-10}, u_{2-5}, u_{2-6}, u_{6-10}\}$
$\psi_6 = \{u_{1-6}, u_{1-10}, u_{2-6}, u_{4-6}, u_{6-10}\}$
$\psi_7 = \{u_{2-8}, u_{1-8}, u_{1-10}, u_{2-5}, u_{2-6}\}$
$\psi_8 = \{u_{2-8}, u_{1-8}, u_{1-10}, u_{2-6}, u_{4-6}\}$
$\psi_9 = \{u_{2-8}, u_{1-8}, u_{2-5}, u_{2-6}, u_{8-12}\}$
$\psi_{10} = \{u_{2-8}, u_{1-8}, u_{2-6}, u_{4-6}, u_{8-12}\}$
$\psi_{11} = \{u_{4-11}, u_{4-6}, u_{6-10}\}$
$\psi_{12} = \{u_{4-11}, u_{5-10}, u_{6-10}\}$
$\psi_{13} = \{u_{1-10}, u_{2-5}, u_{5-10}, u_{6-10}\}$

. . .

Выделение максимального двудольного подграфа H’ из G’

$$a_{1-2} = |\psi_1| + |\psi_2| - |\psi_1 \wedge \psi_2| = 5 + 5 - 4 = 6$$

$$a_{1-3} = |\psi_1| + |\psi_3| - |\psi_1 \wedge \psi_3| = 5 + 5 - 4 = 6$$

$$a_{1-4} = |\psi_1| + |\psi_4| - |\psi_1 \wedge \psi_4| = 5 + 5 - 3 = 7$$

$$a_{1-5} = |\psi_1| + |\psi_5| - |\psi_1 \wedge \psi_5| = 5 + 5 - 4 = 6$$

$$a_{1-6} = |\psi_1| + |\psi_6| - |\psi_1 \wedge \psi_6| = 5 + 5 - 3 = 7$$

$$a_{1-7} = |\psi_1| + |\psi_7| - |\psi_1 \wedge \psi_7| = 5 + 5 - 4 = 6$$

$$a_{1-8} = |\psi_1| + |\psi_8| - |\psi_1 \wedge \psi_8| = 5 + 5 - 3 = 7$$

$$a_{1-9} = |\psi_1| + |\psi_9| - |\psi_1 \wedge \psi_9| = 5 + 5 - 3 = 7$$

$$a_{1-10} = |\psi_1| + |\psi_{10}| - |\psi_1 \wedge \psi_{10}| = 5 + 5 - 2 = 8$$

$$a_{1-11} = |\psi_1| + |\psi_{11}| - |\psi_1 \wedge \psi_{11}| = 5 + 3 - 0 = 8$$

$$a_{1-12} = |\psi_1| + |\psi_{12}| - |\psi_1 \wedge \psi_{12}| = 5 + 3 - 0 = 8$$

$$a_{1-13} = |\psi_1| + |\psi_{13}| - |\psi_1 \wedge \psi_{13}| = 5 + 4 - 2 = 7$$

. . .

$$a_{12-13} = |\psi_{12}| + |\psi_{13}| - |\psi_{12} \wedge \psi_{13}| = 3 + 4 - 2 = 5$$

. . .

Результаты отобразим в матрице:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	6	6	7	6	7	6	7	7	8	8	8	7
2		0	7	6	7	6	7	6	8	7	7	8	8
3			0	6	7	8	7	8	6	7	8	8	8
4				0	8	7	8	7	7	6	7	8	9
5					0	6	7	8	8	9	7	7	6
6						0	8	7	9	8	6	7	7
7							0	6	6	7	8	8	7
8								0	7	6	7	8	8
9									0	6	8	8	8
10										0	7	8	9
11											0	4	6
12												0	5
13													0

$max[a_{i-j}] = a_{4-13}, a_{5-10}, a_{6-9}, a_{11-13} = 9$ дают множества:

$\psi_4 = \{u_{1-6}, u_{1-8}, u_{2-6}, u_{4-6}, u_{8-12}\}$

$\psi_{13} = \{u_{1-10}, u_{2-5}, u_{5-10}, u_{6-10}\}$

Для множеств ψ_9 *и* ψ_{11} *напишем булеву формулу чтобы увидеть только уникальные ребра:*

$\psi_9 \backslash (\psi_4 \vee \psi_{13}) = \{u_{2-8}\}$

$\psi_{11} \backslash (\psi_4 \vee \psi_{13}) = \{u_{4-11}\}$

. . .

В суграфе H, содержащем в себе максимальное число непересекающихся ребер, проведем ребра из ψ_4 и ψ_9 внутри, а ребра из ψ_{11} и ψ_{13} – снаружи: