МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №4.1

по дисциплине: Дискретная математика
тема: «Графы»

Выполнил: ст. группы ПВ-221
Лоёк Никита Викторович

Проверили:
Бондаренко Татьяна Владимировна
Рязанов Юрий Дмитриевич

Белгород 2023 г.

Л а б о р а т о р н а я р а б о т а № 4.1

Цель работы: изучить основные понятия теории графов, способы задания графов, научиться программно реализовывать алгоритмы получения и анализа маршрутов в графах.

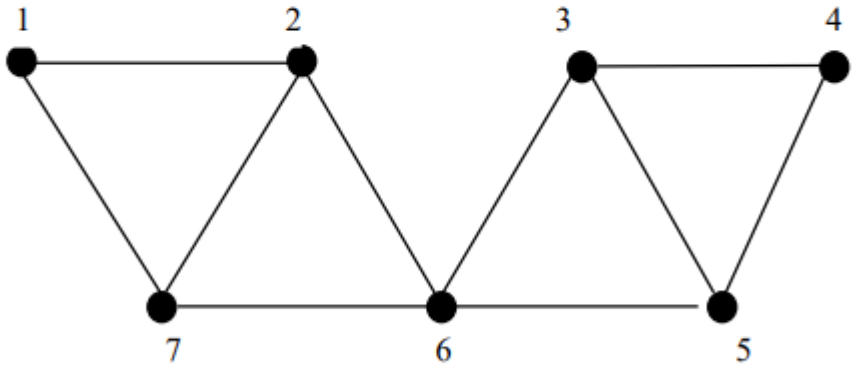
Вариант 12

Вариант 12

а) матрица смежности графа G_1

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

диаграмма графа G_2



б) последовательности вершин

1. (7, 6, 2, 1)
2. (2, 6, 5, 3, 6, 2, 1)
3. (6, 7, 1, 2, 6)
4. (7, 1, 5, 2, 1, 6, 7)
5. (3, 6, 2, 7, 6, 2, 1)

Задания

1. Представить графы G_1 и G_2 (см. ”Варианты заданий”, п.а) матрицей смежности, матрицей инцидентности, диаграммой.

Граф G_1:

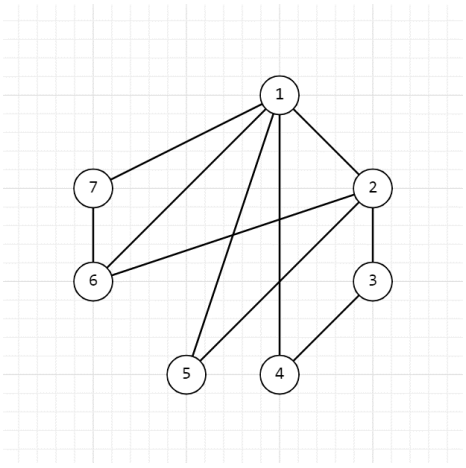
Матрица смежности:

G1	v1	v2	v3	v4	v5	v6	v7
v1	0	1	0	1	1	1	1
v2	1	0	1	0	1	1	0
v3	0	1	0	1	0	0	0
v4	1	0	1	0	0	0	0
v5	1	1	0	0	0	0	0
v6	1	1	0	0	0	0	1
v7	1	0	0	0	0	1	0

Матрица инцидентности:

v1	1	1	1	1	1	0	0	0	0	0
v2	1	0	0	0	0	1	1	1	0	0
v3	0	0	0	0	0	1	0	0	1	0
v4	0	1	0	0	0	0	0	0	1	0
v5	0	0	1	0	0	0	1	0	0	0
v6	0	0	0	1	0	0	0	1	0	1
v7	0	0	0	0	1	0	0	0	0	1

Диаграмма:



Граф G_2:

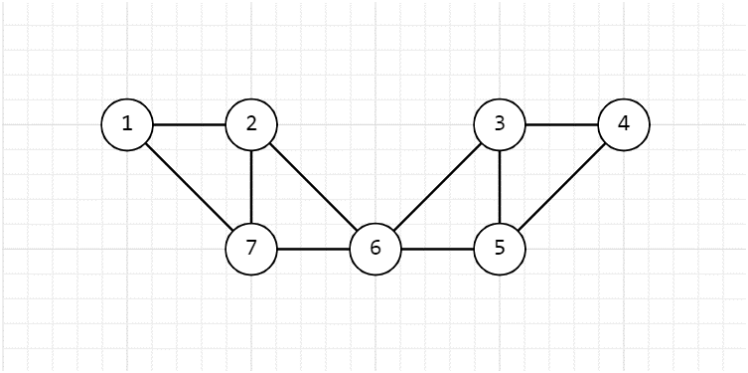
Матрица смежности:

G2	v1	v2	v3	v4	v5	v6	v7
v1	0	1	0	0	0	0	1
v2	1	0	0	0	0	1	1
v3	0	0	0	1	1	1	0
v4	0	0	1	0	1	0	0
v5	0	0	1	1	0	1	0
v6	0	1	1	0	1	0	1
v7	1	1	0	0	0	1	0

Матрица инцидентности:

v1	1	1	0	0	0	0	0	0	0	0
v2	1	0	1	0	0	1	0	0	0	0
v3	0	0	0	1	0	0	1	1	0	0
v4	0	0	0	1	0	0	0	0	1	0
v5	0	0	0	0	0	0	1	0	1	1
v6	0	0	1	0	1	0	0	1	0	1
v7	0	1	0	0	1	1	0	0	0	0

Диаграмма:



2. Определить, являются ли последовательности вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см. ”Варианты заданий”, п.а).

б) последовательности вершин

1. (7, 6, 2, 1)
2. (2, 6, 5, 3, 6, 2, 1)
3. (6, 7, 1, 2, 6)
4. (7, 1, 5, 2, 1, 6, 7)
5. (3, 6, 2, 7, 6, 2, 1)

Последовательность:	Графы:	
(7, 6, 2, 1)	G1:	G2:
Маршрут:	+	+
Цепь:	+	+
Простая цепь:	+	+
Цикл:	-	-
Простой цикл:	-	-

Последовательность:	Графы:	
(2, 6, 5, 3, 6, 2, 1)	G1:	G2:
Маршрут:	-	+
Цепь:	-	-
Простая цепь:	-	-
Цикл:	-	-
Простой цикл:	-	-

Последовательность:	Графы:	
(6, 7, 1, 2, 6)	G1:	G2:
Маршрут:	+	+
Цепь:	+	+
Простая цепь:	-	-
Цикл:	+	+
Простой цикл:	+	+

Последовательность:	Графы:	
(7, 1, 5, 2, 1, 6, 7)	G1:	G2:
Маршрут:	+	-
Цепь:	+	-
Простая цепь:	-	-
Цикл:	+	-
Простой цикл:	-	-

Последовательность:	Графы:	
(3, 6, 2, 7, 6, 2, 1)	G1:	G2:
Маршрут:	-	+
Цепь:	-	-
Простая цепь:	-	-
Цикл:	-	-
Простой цикл:	-	-

3. Написать программу, определяющую, является ли заданная последовательность вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см.”Варианты заданий”, п.а).

Программа:

```
#include <iostream>
#include <vector>


using namespace std;


void matrixOutput(vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            cout << matrix[i][j] << ' ';
        }
        cout << endl;
    }
    cout << endl;
}


void matrixInput(vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            int buf;
            scanf("%d", &buf);
            matrix[i][j] = buf;
        }
    }
}


bool IsRout(vector<vector<bool>> &matrix, vector<int> &rout) {
    for (int i = 0; i < rout.size() - 1; i++) {
        if (!matrix[rout[i] - 1][rout[i + 1] - 1]) {
            return false;
        }
    }

    return true;
}


bool IsChain(vector<vector<bool>> &matrix, vector<int> &rout) {
    vector<vector<int>> chain;
    bool isNotChain = false;
    for (int i = 0; i < rout.size() - 1; i++) {
        isNotChain = false;
        for (int j = 0; j < chain.size(); j++) {
            if (chain[j][0] == rout[i] && chain[j][1] == rout[i + 1] ||
                chain[j][1] == rout[i] && chain[j][0] == rout[i + 1]) {
                isNotChain = true;
            }
        }

        if (!matrix[rout[i] - 1][rout[i + 1] - 1] || isNotChain) {
            return false;
        }
        vector<int> buf = {rout[i], rout[i + 1]};

        chain.push_back(buf);
    }

    return !isNotChain;
}


bool IsPrimeChain(vector<vector<bool>> &matrix, vector<int> &rout) {
    if (!IsRout(matrix, rout)) {
        return false;
    }

    for (int i = 0; i < rout.size() - 1; i++) {
        for (int j = i + 1; j < rout.size(); j++) {
```

```
            if (rout[i] == rout[j]) {
                return false;
            }
        }
    }

    return true;
}


bool IsCycle(vector<vector<bool>> &matrix, vector<int> &rout) {
    if (rout[0] != rout[rout.size() - 1]) {
        return false;
    }

    if (!IsChain(matrix, rout)) {
        return false;
    }

    return true;
}


bool IsPrimeCycle(vector<vector<bool>> &matrix, vector<int> &rout) {
    if (rout[0] != rout[rout.size() - 1]) {
        return false;
    }

    for (int i = 0; i < rout.size() - 1; i++) {
        for (int j = i + 1; j < rout.size(); j++) {
            if (i == 0 && j == rout.size() - 1) {
                continue;
            } else if (rout[i] == rout[j]) {
                return false;
            }
        }
    }

    return true;
}


void outputVector(vector<int> &v) {
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
}


void outputSequenceProperty(vector<vector<bool>> &matrix, vector<int> &sequence) {
    cout << "For ";
    outputVector(sequence);
    cout << "IsRout: " << IsRout(matrix, sequence) << endl;
    cout << "IsChain: " << IsChain(matrix, sequence) << endl;
    cout << "IsPrimeChain: " << IsPrimeChain(matrix, sequence) << endl;
    cout << "IsCycle: " << IsCycle(matrix, sequence) << endl;
    cout << "IsPrimeCycle: " << IsPrimeCycle(matrix, sequence) << endl << endl;


int main() {
    vector<vector<bool>> matrix(7, vector<bool>(7, false));
    matrixInput(matrix);

    vector<int> rout_1 = {7, 6, 2, 1};
    vector<int> rout_2 = {2, 6, 5, 3, 6, 2, 1};
    vector<int> rout_3 = {6, 7, 1, 2, 6};
    vector<int> rout_4 = {7, 1, 5, 2, 1, 6, 7};
    vector<int> rout_5 = {3, 6, 2, 7, 6, 2, 1};

    outputSequenceProperty(matrix, rout_1);
    outputSequenceProperty(matrix, rout_2);
    outputSequenceProperty(matrix, rout_3);
    outputSequenceProperty(matrix, rout_4);
    outputSequenceProperty(matrix, rout_5);

    return 0;
}
```

Вывод программы:

G1:

```
0   1   0   1   1   1   1
1   0   1   0   1   1   0
0   1   0   1   0   0   0
1   0   1   0   0   0   0
1   1   0   0   0   0   0
1   1   0   0   0   0   1
1   0   0   0   0   1   0
For 7 6 2 1
IsRout: 1
IsChain: 1
IsPrimeChain: 1
IsCycle: 0
IsPrimeCycle: 0

For 2 6 5 3 6 2 1
IsRout: 0
IsChain: 0
IsPrimeChain: 0
IsCycle: 0
IsPrimeCycle: 0

For 6 7 1 2 6
IsRout: 1
IsChain: 1
IsPrimeChain: 0
IsCycle: 1
IsPrimeCycle: 1

For 7 1 5 2 1 6 7
IsRout: 1
IsChain: 1
IsPrimeChain: 0
IsCycle: 1
IsPrimeCycle: 0

For 3 6 2 7 6 2 1
IsRout: 0
IsChain: 0
IsPrimeChain: 0
IsCycle: 0
IsPrimeCycle: 0


Process finished with exit code 0
```

G2:

```
0   1   0   0   0   0   1
1   0   0   0   0   1   1
0   0   0   1   1   1   0
0   0   1   0   1   0   0
0   0   1   1   0   1   0
0   1   1   0   1   0   1
1   1   0   0   0   1   0
For 7 6 2 1
IsRout: 1
IsChain: 1
IsPrimeChain: 1
IsCycle: 0
IsPrimeCycle: 0

For 2 6 5 3 6 2 1
IsRout: 1
IsChain: 0
IsPrimeChain: 0
IsCycle: 0
IsPrimeCycle: 0

For 6 7 1 2 6
IsRout: 1
IsChain: 1
IsPrimeChain: 0
IsCycle: 1
IsPrimeCycle: 1

For 7 1 5 2 1 6 7
IsRout: 0
IsChain: 0
IsPrimeChain: 0
IsCycle: 0
IsPrimeCycle: 0

For 3 6 2 7 6 2 1
IsRout: 1
IsChain: 0
IsPrimeChain: 0
IsCycle: 0
IsPrimeCycle: 0


Process finished with exit code 0
```

4. Написать программу, получающую все маршруты заданной длины, выходящие из заданной вершины. Использовать программу для получения всех маршрутов заданной длины в графах G1 и G2 (см. ”Варианты заданий”, п.а).

Программа:

```
#include <iostream>
#include <vector>


using namespace std;


void matrixInput(vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            int buf;
            scanf("%d", &buf);
            matrix[i][j] = buf;
        }
    }
}


void outputVector(vector<int> &v) {
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
}


void outputVertex(vector<vector<int>> &v) {
    for (int i = 0; i < v.size(); i++) {
        outputVector(v[i]);
    }
}


void GetRoutes_(vector<vector<bool>> &matrix, int maxLen, vector<int> G, vector<vector<int>>
&res) {
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i][G[G.size() - 1] - 1]) {
            G.push_back(i + 1);
            if (G.size() - 1 == maxLen) {
                res.push_back(G);
            } else {
                GetRoutes_(matrix, maxLen, G, res);
            }
            G.pop_back();
        }
    }
}


vector<vector<int>> GetRoutes(vector<vector<bool>> &matrix, int v, int maxLen) {
    vector<vector<int>> routs = {};
    GetRoutes_(matrix, maxLen, vector<int>(1, v), routs);

    return routs;
}


vector<vector<int>> GetAllRoutes(vector<vector<bool>> &matrix, int maxLen) {
    vector<vector<int>> res = {};

    for (int i = 1; i <= matrix.size(); i++) {
        vector<vector<int>> buf = GetRoutes(matrix, i, maxLen);
        res.insert(res.end(), buf.begin(), buf.end());
    }

    return res;
}


int main() {
    vector<vector<bool>> matrix(7, vector<bool>(7, false));
```

```
    matrixInput(matrix);

    vector<vector<int>> v = GetRoutes(matrix, 1, 2);
    outputVertex(v);

    return 0;
}
```

Вывод программы:

G1:

```
0 1 0 1 1 1 1
1 0 1 0 1 1 0
0 1 0 1 0 0 0
1 0 1 0 0 0 0
1 1 0 0 0 0 0
1 1 0 0 0 0 1
1 0 0 0 0 1 0
1 2 1
1 2 3
1 2 5
1 2 6
1 4 1
1 4 3
1 5 1
1 5 2
1 6 1
1 6 2
1 6 7
1 7 1
1 7 6

Process finished with exit code 0
```

G2:

```
0 1 0 0 0 0 1
1 0 0 0 0 1 1
0 0 0 1 1 1 0
0 0 1 0 1 0 0
0 0 1 1 0 1 0
0 1 1 0 1 0 1
1 1 0 0 0 1 0
1 2 1
1 2 6
1 2 7
1 7 1
1 7 2
1 7 6

Process finished with exit code 0
```

5. Написать программу, определяющую количество маршрутов заданной длины между каждой парой вершин графа. Использовать программу для определения количества маршрутов заданной длины между каждой парой вершин в графах G1 и G2 (см. ”Варианты заданий”, п.а).

Программа:

```
#include <iostream>
#include <vector>


using namespace std;


void matrixInput(vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            int buf;
            scanf("%d", &buf);
            matrix[i][j] = buf;
        }
    }
}


void outputSumRoutes(vector<vector<int>> &v) {
    for (int i = 0; i < v.size(); i++) {
        for (int j = 0; j < v[i].size(); j++) {
            if (v[i].size() == 1) {
                cout << ": " << v[i][j] << endl;
            } else {
                cout << v[i][j] << " ";
            }
        }
    }
}


int GetSumRoutes_(vector<vector<bool>> &matrix, int maxLen, vector<int> G, int vEnd) {
    int sumRoutes = 0;
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i][G[G.size() - 1] - 1]) {
            G.push_back(i + 1);
            if (G.size() - 1 == maxLen && vEnd == i + 1) {
                return 1;
            } else if (G.size() - 1 != maxLen) {
                sumRoutes += GetSumRoutes_(matrix, maxLen, G, vEnd);
            }
            G.pop_back();
        }
    }
    return sumRoutes;
}


int GetSumRoutes(vector<vector<bool>> &matrix, int v, int maxLen, int vEnd) {
    int sumRoutes = GetSumRoutes(matrix, maxLen, vector<int>(1, v), vEnd);

    return sumRoutes;
}


vector<vector<int>> GetAllSumRoutes(vector<vector<bool>> &matrix, int maxLen) {
    vector<vector<int>> res = {};

    for (int i = 1; i <= matrix.size(); i++) {
        for (int j = 1; j <= matrix.size(); j++) {
            int buf = GetSumRoutes(matrix, i, maxLen, j);

            res.push_back(vector<int>{i, j});
            res.push_back(vector<int>{buf});
        }
    }

    return res;
}
```

```
int main() {
    vector<vector<bool>> matrix(7, vector<bool>(7, false));
    matrixInput(matrix);

    cout << "5 -> 1 : " << GetSumRoutes(matrix, 5, 3, 1) << endl;

    return 0;
}
```

Вывод программы:

G1:

```
0   1   0   1   1   1   1
1   0   1   0   1   1   0
0   1   0   1   0   0   0
1   0   1   0   0   0   0
1   1   0   0   0   0   0
1   1   0   0   0   0   1
1   0   0   0   0   1   0
5 -> 1 : 7

Process finished with exit code 0
```

G2:

```
0   1   0   0   0   0   1
1   0   0   0   0   1   1
0   0   0   1   1   1   0
0   0   1   0   1   0   0
0   0   1   1   0   1   0
0   1   1   0   1   0   1
1   1   0   0   0   1   0
5 -> 1 : 2

Process finished with exit code 0
```

6. Написать программу, определяющую все маршруты заданной длины между заданной парой вершин графа. Использовать программу для определения всех маршрутов заданной длины между заданной парой вершин в графах G1 и G2 (см. ”Варианты заданий”, п.а).

Программа:

```
#include <iostream>
#include <vector>


using namespace std;


void matrixInput(vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            int buf;
            scanf("%d", &buf);
            matrix[i][j] = buf;
        }
    }
}


void outputVector(vector<int> &v) {
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
}


void outputVertex(vector<vector<int>> &v) {
    for (int i = 0; i < v.size(); i++) {
        outputVector(v[i]);
    }
}


void GetPairRoutes_(vector<vector<bool>> &matrix, int maxLen, vector<int> G, int vEnd,
vector<vector<int>> &res) {
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i][G[G.size() - 1] - 1]) {
            G.push_back(i + 1);
            if (G.size() - 1 == maxLen && vEnd == (i + 1)) {
                res.push_back(G);
            } else if (G.size() - 1 != maxLen) {
                GetPairRoutes_(matrix, maxLen, G, vEnd, res);
            }
            G.pop_back();
        }
    }
}


vector<vector<int>> GetPairRoutes(vector<vector<bool>> &matrix, int v, int maxLen, int vEnd) {
    vector<vector<int>> routs = {};
    GetPairRoutes_(matrix, maxLen, vector<int>(1, v), vEnd, routs);

    return routs;
}


vector<vector<int>> GetAllPairRoutes(vector<vector<bool>> &matrix, int maxLen) {

    vector<vector<int>> res = {};

    for (int i = 1; i <= matrix.size(); i++) {
        for (int j = 1; j <= matrix.size(); j++) {
            vector<vector<int>> buf = GetPairRoutes(matrix, i, maxLen, j);
            res.insert(res.end(), buf.begin(), buf.end());
        }
    }
```

```
    return res;
}


int main() {
    vector<vector<bool>> matrix(7, vector<bool>(7, false));
    matrixInput(matrix);

    vector<vector<int>> vertexArray = GetPairRoutes(matrix, 5, 3, 1);
    outputVertex(vertexArray);

    return 0;
}
```

Вывод программы:

G1:

```
0   1   0   1   1   1   1
1   0   1   0   1   1   0
0   1   0   1   0   0   0
1   0   1   0   0   0   0
1   1   0   0   0   0   0
1   1   0   0   0   0   1
1   0   0   0   0   1   0
5 1 2 1
5 1 4 1
5 1 5 1
5 1 6 1
5 1 7 1
5 2 5 1
5 2 6 1

Process finished with exit code 0
```

G2:

```
0   1   0   0   0   0   1
1   0   0   0   0   1   1
0   0   0   1   1   1   0
0   0   1   0   1   0   0
0   0   1   1   0   1   0
0   1   1   0   1   0   1
1   1   0   0   0   1   0
5 6 2 1
5 6 7 1

Process finished with exit code 0
```

7. Написать программу, получающую все простые максимальные цепи, выходящие из заданной вершины графа. Использовать программу для получения всех простые максимальных цепей, выходящих из заданной вершины в графах G1 и G2 (см. ”Варианты заданий”, п.а).

Программа:

```
#include <iostream>
#include <vector>


using namespace std;


void matrixInput(vector<vector<bool>> &matrix) {
    for (int i = 0; i < matrix.size(); i++) {
        for (int j = 0; j < matrix.size(); j++) {
            int buf;
            scanf("%d", &buf);
            matrix[i][j] = buf;
        }
    }
}


void outputVector(vector<int> &v) {
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
}


void outputVertex(vector<vector<int>> &v) {
    for (int i = 0; i < v.size(); i++) {
        outputVector(v[i]);
    }
}


void GetMaxChains_(vector<vector<bool>> &matrix, vector<int> G, vector<vector<int>> &res) {
    bool areNoWaysToAnyPoint = true;
    for (int i = 0; i < matrix.size() && G.size() <= matrix.size(); i++) {
        if (matrix[i][G[G.size() - 1] - 1]) {
            bool isNoPointInG = true;
            for (int j = 0; j < G.size() && isNoPointInG; j++) {
                if (G[j] == (i + 1)) {
                    isNoPointInG = false;
                }
            }

            areNoWaysToAnyPoint = areNoWaysToAnyPoint && !isNoPointInG;

            if (isNoPointInG) {
                G.push_back(i + 1);
                GetMaxChains_(matrix, G, res);
                G.pop_back();
            }
        }
    }

    if (areNoWaysToAnyPoint) {
        res.push_back(G);
    }
}


vector<vector<int>> GetMaxChains(vector<vector<bool>> &matrix, int v) {
    vector<vector<int>> res = {};
    GetMaxChains_(matrix, vector<int>(1, v), res);

    return res;
}
```

```cpp
vector<vector<int>> GetAllMaxChains(vector<vector<bool>> &matrix) {
    vector<vector<int>> res = {};

    for (int i = 1; i <= matrix.size(); i++) {
        vector<vector<int>> buf = GetMaxChains(matrix, i);
        res.insert(res.end(), buf.begin(), buf.end());
    }

    return res;
}


int main() {
    vector<vector<bool>> matrix(7, vector<bool>(7, false));
    matrixInput(matrix);

    vector<vector<int>> vertexArray = GetMaxChains(matrix, 1);
    outputVertex(vertexArray);

    return 0;
}
```

Вывод программы:

G1:

```
0   1   0   1   1   1   1
1   0   1   0   1   1   0
0   1   0   1   0   0   0
1   0   1   0   0   0   0
1   1   0   0   0   0   0
1   1   0   0   0   0   1
1   0   0   0   0   1   0
1 2 3 4
1 2 5
1 2 6 7
1 4 3 2 5
1 4 3 2 6 7
1 5 2 3 4
1 5 2 6 7
1 6 2 3 4
1 6 2 5
1 6 7
1 7 6 2 3 4
1 7 6 2 5

Process finished with exit code 0
```

G2:

```
0   1   0   0   0   0   1
1   0   0   0   0   1   1
0   0   0   1   1   1   0
0   0   1   0   1   0   0
0   0   1   1   0   1   0
0   1   1   0   1   0   1
1   1   0   0   0   1   0
1 2 6 3 4 5
1 2 6 3 5 4
1 2 6 5 3 4
1 2 6 5 4 3
1 2 6 7
1 2 7 6 3 4 5
1 2 7 6 3 5 4
1 2 7 6 5 3 4
1 2 7 6 5 4 3
1 7 2 6 3 4 5
1 7 2 6 3 5 4
1 7 2 6 5 3 4
1 7 2 6 5 4 3
1 7 6 2
1 7 6 3 4 5
1 7 6 3 5 4
1 7 6 5 3 4
1 7 6 5 4 3

Process finished with exit code 0
```

Вывод

Вывод: в ходе работы я изучил основные понятия теории графов, способы задания графов, научился программно реализовывать алгоритмы получения и анализа маршрутов в графах.