Отчёт по расчётной работе по дисциплине ПиОИвИС

Тема: Графы

Цель

Научится работать и проводить различные операции с графами.

Задача

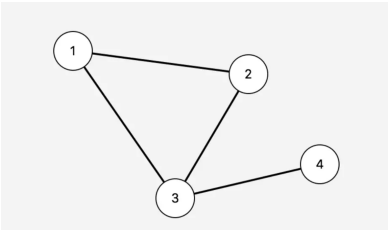
Найти мосты в неориентированном графе.

Вариант

5.15 (матрица инцидентности)

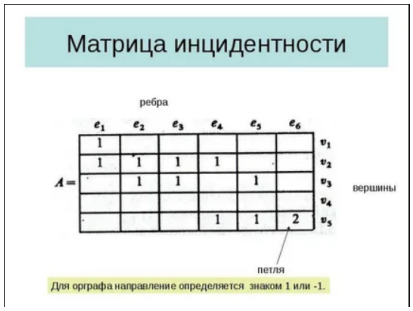
Список ключевых понятий (определения)

- **Граф** — математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин.
- **Неориентированный граф** — это граф, где рёбра не имеют направления, что делает связь между вершинами двусторонней.



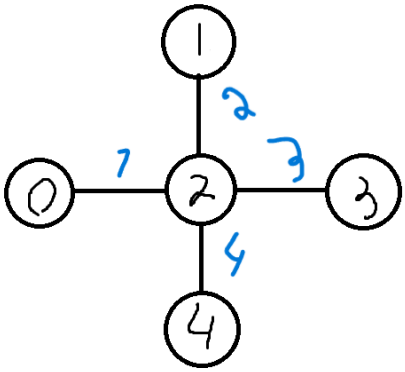
- **Инцидентность** — понятие, используемое только в отношении ребра и вершины. Две вершины или два ребра не могут быть инцидентны.
- **Матрица инцидентности** — одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро и вершина). Столбцы матрицы соответствуют рёбрам, строки — вершинам. Ненулевое

значение в ячейке матрицы указывает связь между вершиной и ребром (их инцидентность).

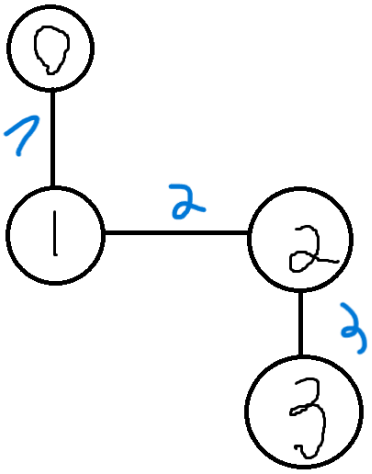


Приведённые примеры неориентированных графов

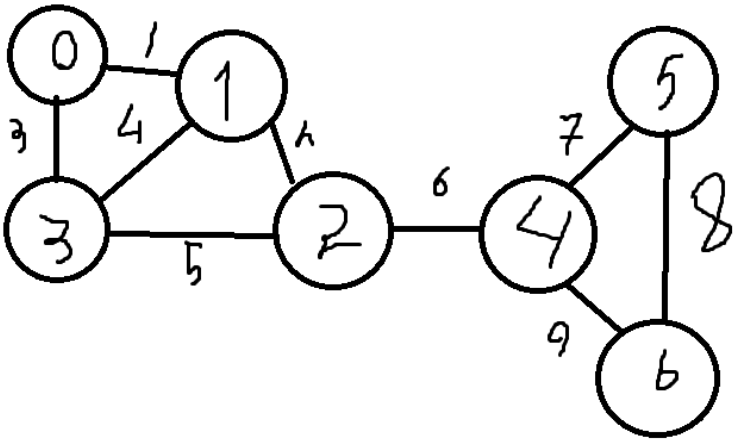
- Граф 1



- Граф 2



- Граф 3



Алгоритм

1. Пользователь вводит количество вершин и рёбер графа.
2. Затем вводится матрица инцидентности, где строки представляют вершины, а столбцы — рёбра.
3. Функция matrixToAdjList преобразует матрицу инцидентности в список смежности.
4. Используется DFS для обхода графа.
5. После завершения DFS алгоритм выводит все найденные мосты в графе.

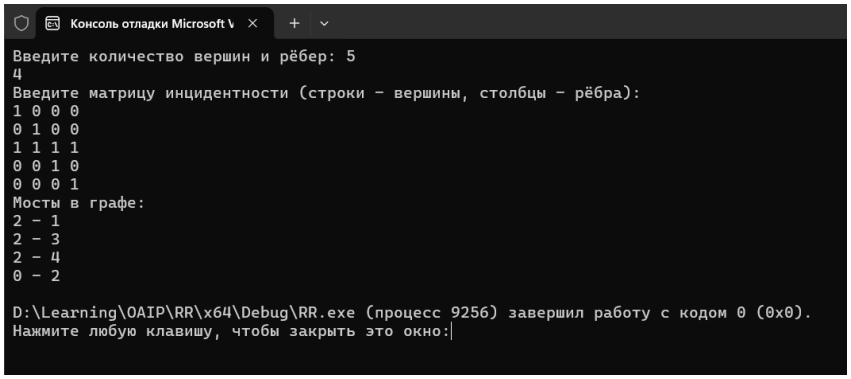
Код

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

using namespace std;

//                                                    (DFS)

void dfs(int v, int parent, vector<vector<int>>& adj, vector<int>&
    visited, vector<int>& tin, vector<int>& low, int& timer,
   vector<pair<int, int>>& bridges) {
    visited[v] = 1;
    tin[v] = low[v] = timer++;

    for (int to : adj[v]) {
        if (to == parent) continue; //

        if (visited[to]) {
```

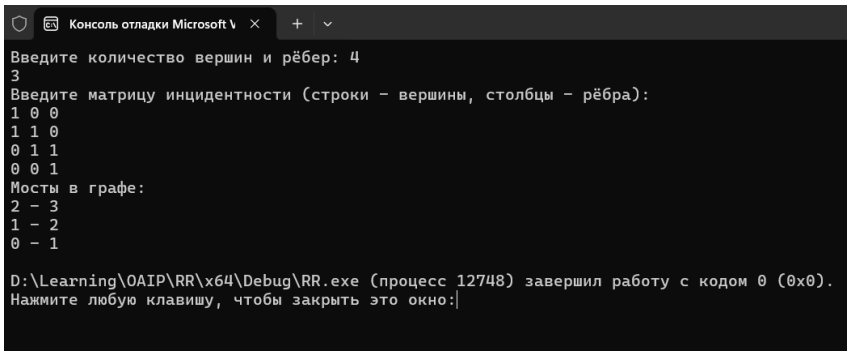
```
            //                                          low,          "
               to"
            low[v] = min(low[v], tin[to]);
        }
        else {
            dfs(to, v, adj, visited, tin, low, timer, bridges);
            low[v] = min(low[v], low[to]);
            if (low[to] > tin[v]) {
                bridges.emplace_back(v, to);
            }
        }
    }
}

//

vector<vector<int>> matrixToAdjList(const vector<vector<int>>&
   incMatrix) {
    int n = incMatrix.size(); //
    int m = incMatrix[0].size(); //
    vector<vector<int>> adj(n);

    for (int j = 0; j < m; ++j) {
        int u = -1, v = -1;
        for (int i = 0; i < n; ++i) {
            if (incMatrix[i][j] == 1) {
                if (u == -1) u = i;
                else v = i;
            }
        }
        if (u != -1 && v != -1) {
            adj[u].push_back(v);
            adj[v].push_back(u);
        }
    }
    return adj;
}

int main() {
    int n, m;
    setlocale(LC_ALL, "Rus");
    cout << "              ␣                    ␣             ␣  ␣
                 :␣";
    cin >> n >> m;

    vector<vector<int>> incMatrix(n, vector<int>(m));
    cout << "              ␣              ␣
                                ␣(             ␣-␣              ,␣
                      ␣-␣          ):\n";
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
```

```
            cin >> incMatrix[i][j];
        }
    }

    //


    vector<vector<int>> adj = matrixToAdjList(incMatrix);

    //

    vector<int> visited(n, 0), tin(n, -1), low(n, -1);
    vector<pair<int, int>> bridges;
    int timer = 0;

    //                      DFS
    for (int i = 0; i < n; ++i) {
        if (!visited[i]) {
            dfs(i, -1, adj, visited, tin, low, timer, bridges);
        }
    }

    //
    cout << "           ␣  ␣           :\n";
    for (const auto& bridge : bridges) {
        cout << bridge.first << "␣-␣" << bridge.second << "\n";
    }

    return 0;
}
```

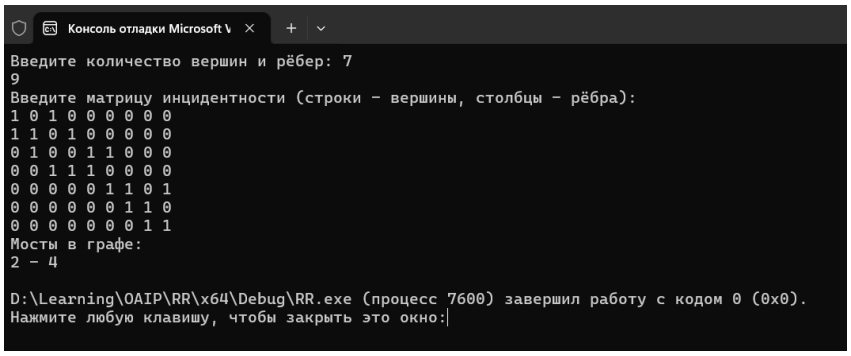
Пример работы кода с Графом №1



Пример работы кода с Графом №2



Пример работы кода с Графом №3



Вывод

В результате выполнения данной работы были получены следующие практические навыки:

- Изучены основы теории графов.
- Изучены способы представления графов.
- Изучены базовые алгоритмы для работы с графами.