

Отчёт по расчётной работе по дисциплине ПиОИВИС

Тема: Графы

Цель

Научится работать и проводить различные операции с графами.

Задача

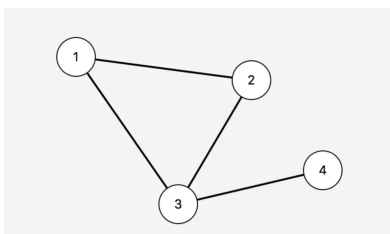
Найти мосты в неориентированном графе.

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).

Матрица инцидентности

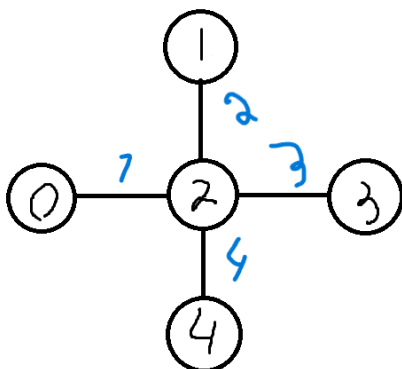
	ребра						
	e_1	e_2	e_3	e_4	e_5	e_6	
$A =$	1						v_1
	1	1	1	1			v_2
		1	1		1		v_3
							v_4
				1	1	2	v_5

петля

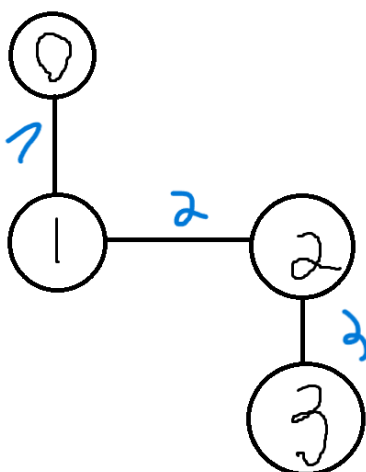
Для орграфа направление определяется знаком 1 или -1.

Приведённые примеры неориентированных графов

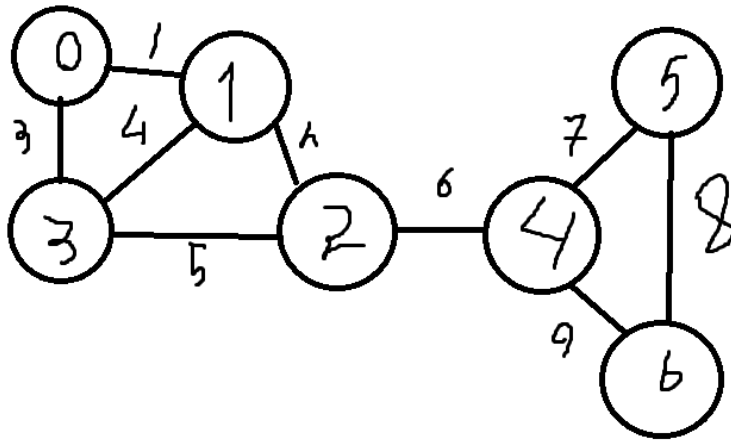
- Граф 1



- 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

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  // (DFS)
8
9  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) {
10     visited[v] = 1;
11     tin[v] = low[v] = timer++;
12
13     for (int to : adj[v]) {
14         if (to == parent) continue; //
15
16         if (visited[to]) {

```

```

16         //                                     low,         "
17         to"
18         low[v] = min(low[v], tin[to]);
19     }
20     else {
21         dfs(to, v, adj, visited, tin, low, timer, bridges);
22         low[v] = min(low[v], low[to]);
23         if (low[to] > tin[v]) {
24             bridges.emplace_back(v, to);
25         }
26     }
27 }
28
29 //
30 vector<vector<int>> matrixToAdjList(const vector<vector<int>>&
31     incMatrix) {
32     int n = incMatrix.size(); //
33     int m = incMatrix[0].size(); //
34     vector<vector<int>> adj(n);
35
36     for (int j = 0; j < m; ++j) {
37         int u = -1, v = -1;
38         for (int i = 0; i < n; ++i) {
39             if (incMatrix[i][j] == 1) {
40                 if (u == -1) u = i;
41                 else v = i;
42             }
43         }
44         if (u != -1 && v != -1) {
45             adj[u].push_back(v);
46             adj[v].push_back(u);
47         }
48     }
49     return adj;
50 }
51
52 int main() {
53     int n, m;
54     setlocale(LC_ALL, "Rus");
55     cout << "          □          □ □
56             :□";
57     cin >> n >> m;
58
59     vector<vector<int>> incMatrix(n, vector<int>(m));
60     cout << "          □          □
61             □(          □-□ ,□
62             □-□          ):\n";
63     for (int i = 0; i < n; ++i) {
64         for (int j = 0; j < m; ++j) {

```

```

61         cin >> incMatrix[i][j];
62     }
63 }
64
65 //
66
67 vector<vector<int>> adj = matrixToAdjList(incMatrix);
68
69 //
70 vector<int> visited(n, 0), tin(n, -1), low(n, -1);
71 vector<pair<int, int>> bridges;
72 int timer = 0;
73
74 // DFS
75 for (int i = 0; i < n; ++i) {
76     if (!visited[i]) {
77         dfs(i, -1, adj, visited, tin, low, timer, bridges);
78     }
79 }
80
81 //
82 cout << "          □ □          :\\n";
83 for (const auto& bridge : bridges) {
84     cout << bridge.first << "□-□" << bridge.second << "\\n";
85 }
86
87 return 0;
88 }

```

Пример работы кода с Графом №1

```

Консоль отладки Microsoft V  x  +  v
Введите количество вершин и рёбер: 5
4
Введите матрицу инцидентности (строки - вершины, столбцы - рёбра):
1 0 0 0
0 1 0 0
0 1 1 1
0 0 1 0
0 0 0 1
Мосты в графе:
2 - 1
2 - 3
2 - 4
0 - 2
D:\Learning\OAIP\RR\x64\Debug\RR.exe (процесс 9256) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:

```

Пример работы кода с Графом №2

```
Консоль отладки Microsoft V x + v
Введите количество вершин и рёбер: 4
3
Введите матрицу инцидентности (строки – вершины, столбцы – рёбра):
1 0 0
1 1 0
0 1 1
0 0 1
Мосты в графе:
2 - 3
1 - 2
0 - 1
D:\Learning\OAIP\RR\x64\Debug\RR.exe (процесс 12748) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

Пример работы кода с Графом №3

```
Консоль отладки Microsoft V x + v
Введите количество вершин и рёбер: 7
9
Введите матрицу инцидентности (строки – вершины, столбцы – рёбра):
1 0 1 0 0 0 0 0 0
1 1 0 1 0 0 0 0 0
0 1 0 0 1 1 0 0 0
0 0 1 1 1 0 0 0 0
0 0 0 0 0 1 1 0 1
0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 1 1
Мосты в графе:
2 - 4
D:\Learning\OAIP\RR\x64\Debug\RR.exe (процесс 7600) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

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.