

Отчёт по расчётной работе по дисциплине ПиОИВИС

Графы

Цель

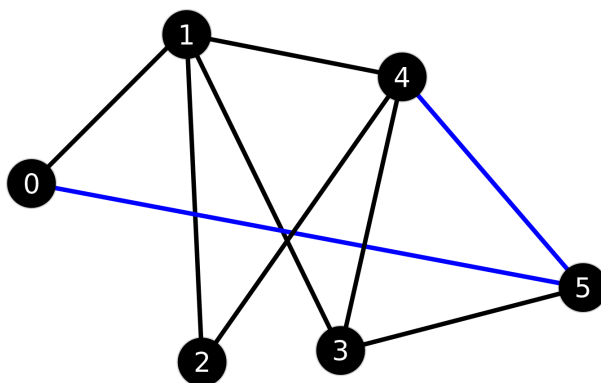
Найти объединение множества неориентированных графов

Вариант

4.8 мс

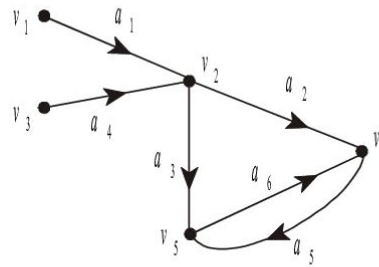
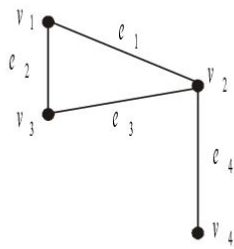
Определения

- **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 у которого рёбра не указывают направление. Это значит, что из любой вершины можно попасть в любую точку графа.



- **Смежность** — непосредственная близость, примыкание. В теории графов смежность вершин соответствует наличию ребра между ними.
- **Матрица смежности** - это вид представления графа в виде матрицы, когда пересечение столбцов и строк задаёт дуги.

Матричные представления графа



Матрица смежности

v_1	v_2	v_3	v_4	
0	1	1	0	v_1
1	0	1	1	v_2
1	1	0	0	v_3
0	1	0	0	v_4

v_1	v_2	v_3	v_4	v_5	
0	1	0	0	0	v_1
0	0	0	1	1	v_2
0	1	0	0	0	v_3
0	0	0	0	1	v_4
0	0	0	1	0	v_5

Алгоритм

1. Создаётся пустой граф `unionGraph` для объединения графов.
2. Ввод первого графа.
3. Ввод второго графа.
4. Подгон размера матриц с помощью `resizeMatrix`.
5. Объединение графов функцией `UnionGraphs()`.
6. Считать первый и второй граф из файла `graph1.txt` и `graph2.txt` с помощью `readGraphFromFile()` и объединить его с `unionGraph`.
7. Используя `displayGraph()` вывести итоговую матрицу смежности объединенного графа.

Код

```

1 #include <iostream>
2 #include <vector>
3 #include <fstream>
4 using namespace std;
5
6 class Graph {
7 private:
8     vector<vector<int>> Matrix_smezh;
9 public:

```

```

10 Graph(int razmer = 0) {
11     Matrix_smezh.resize(razmer, vector<int>(razmer, 0));
12 }
13
14 void addRebro(int v1, int v2, int ves) {
15     if (v1 < Matrix_smezh.size() && v2 < Matrix_smezh.size())
16     {
17         Matrix_smezh[v1][v2] = ves;
18         Matrix_smezh[v2][v1] = ves; //
19     }
20 }
21
22 void UnionGraphs(const Graph& drugoy) {
23     int maxSize = max(Matrix_smezh.size(), drugoy.Matrix_smezh
24         .size());
25     resizeMatrix(maxSize);
26     for (int i = 0; i < drugoy.Matrix_smezh.size(); ++i) {
27         for (int j = 0; j < drugoy.Matrix_smezh[i].size(); ++j
28             ) {
29             if (drugoy.Matrix_smezh[i][j]) {
30                 Matrix_smezh[i][j] = drugoy.Matrix_smezh[i][j
31                     ];
32             }
33         }
34     }
35 }
36
37 void displayGraph() const {
38     cout << "Matrica Smejnosti:" << endl;
39     for (const auto& row : Matrix_smezh) {
40         for (int val : row) {
41             cout << val << " ";
42         }
43         cout << endl;
44     }
45 }
46
47 void resizeMatrix(int newSize) {
48     Matrix_smezh.resize(newSize);
49     for (auto& row : Matrix_smezh) {
50         row.resize(newSize, 0);
51     }
52 }
53
54 const vector<vector<int>>& getMatrix_smezh() const {
55     return Matrix_smezh;
56 }
57
58 };

```

```

55 void writeGraphToFile(const Graph& graph, const string& filename)
56 {
57     ofstream file(filename);
58     if (!file.is_open()) {
59         cout << "Ne_ydalos_otkrit_file:_ " << filename << endl;
60         return;
61     }
62     const auto& Matrix_smezh = graph.getMatrix_smezh();
63     file << Matrix_smezh.size() << endl;
64     for (const auto& row : Matrix_smezh) {
65         for (int val : row) {
66             file << val << " ";
67         }
68         file << endl;
69     }
70     file.close();
71     cout << "Graph_zapisan_v_file_" << filename << endl;
72 }
73
74 void readGraphFromFile(Graph& graph, const string& filename) {
75     ifstream file(filename);
76     if (!file.is_open()) {
77         cout << "Ne_ydalos_otkrit_file:_ " << filename << endl;
78         return;
79     }
80     int size;
81     file >> size;
82     Graph tempGraph(size);
83     for (int i = 0; i < size; ++i) {
84         for (int j = 0; j < size; ++j) {
85             int edge;
86             file >> edge;
87             tempGraph.addRebro(i, j, edge);
88         }
89     }
90     graph.UnionGraphs(tempGraph);
91     file.close();
92 }
93
94 Graph inputGraph() {
95     int size;
96     cout << "Vvedite_razmer_matrici_smejnosti:_ ";
97     cin >> size;
98     Graph graph(size);
99     cout << "Vvedite_matricy_smejnosty_(po_ " << size << "chisel_v_
100         _kajdoi_stroke):" << endl;
101     for (int i = 0; i < size; ++i) {
102         for (int j = 0; j < size; ++j) {
103             int rebro;
104             cin >> rebro;
105             graph.addRebro(i, j, rebro);

```

```

104     }
105 }
106 return graph;
107 }
108
109 int main() {
110     Graph unionGraph;
111     char vybor;
112     setlocale(0, "");
113     cout << "Vvesti novie graphs (y/n): ";
114     cin >> vybor;
115     if (vybor == 'y' || vybor == 'Y') {
116
117         cout << "Vvedite dannie dlya pervogo grapha." << endl;
118         Graph g1 = inputGraph();
119         cout << "Save graph 1 in file (y/n): ";
120         cin >> vybor;
121         if (vybor == 'y' || vybor == 'Y') {
122             string filename;
123             cout << "Vvedite imya file: ";
124             cin >> filename;
125             if (filename.substr(filename.size() - 4) != ".txt") {
126                 filename += ".txt";
127             }
128             writeGraphToFile(g1, filename);
129         }
130
131         cout << "Vvedite dannie dlya pervogo grapha." << endl;
132         Graph g2 = inputGraph();
133         cout << "Save graph 2 in file (y/n): ";
134         cin >> vybor;
135         if (vybor == 'y' || vybor == 'Y') {
136             string filename;
137             cout << "Vvedite imya file dlya file 2: ";
138             cin >> filename;
139             if (filename.substr(filename.size() - 4) != ".txt") {
140                 filename += ".txt";
141             }
142             writeGraphToFile(g2, filename);
143         }
144
145         unionGraph.UnionGraphs(g1);
146         unionGraph.UnionGraphs(g2);
147     }
148     else {
149
150         readGraphFromFile(unionGraph, "graph1.txt");
151         readGraphFromFile(unionGraph, "graph2.txt");
152     }
153     cout << "Result:" << endl;
154     unionGraph.displayGraph();

```

```
155     return 0;  
156 }
```

Пример работы кода

```
Введите данные для первого графа.  
Введите размер матрицы смежности: 3  
Введите матрицу смежности (по 3 чисел в каждой строке):  
1 0 1  
0 0 1  
1 1 1  
Хотите сохранить первый граф в файл? (y/n): n  
Введите данные для второго графа.  
Введите размер матрицы смежности: 3  
Введите матрицу смежности (по 3 чисел в каждой строке):  
1 1 0  
1 0 1  
0 1 0  
Хотите сохранить второй граф в файл? (y/n): n  
Объединенный граф:  
Матрица смежности:  
1 1 1  
1 0 1  
1 1 1
```

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.