Отчёт по расчётной работе по дисциплине ПиОИвИС

Графы

Задача

Научится работать и проводить различные операции с графами.

Цель

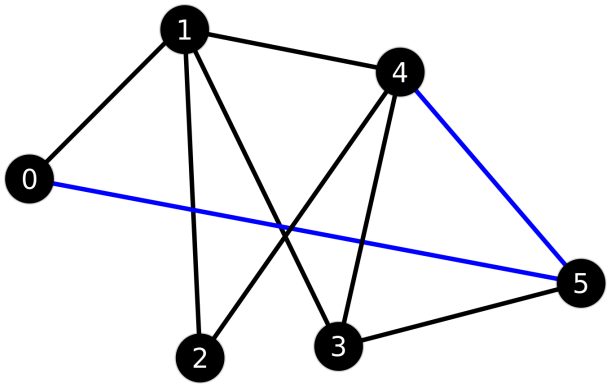
Найти объединение множества неориентированных графов

Вариант

4.8 мс

Определения

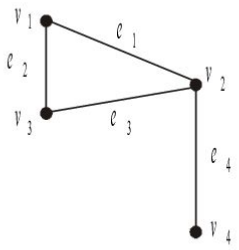
- **Граф** — математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством вершин, и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин.
- **Неориентрованный граф** — это граф у которого рёбра не указывают направление. Это значит, что из любой вершины можно попасть в любую точку графа.

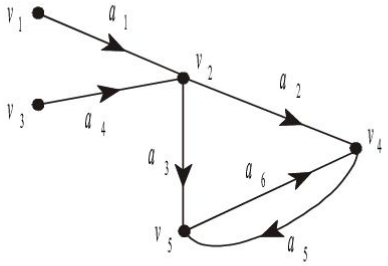


- **Смежность** — непосредственная близость, примыкание. В теории графов смежность вершин соответствует наличию ребра между ними.

- **Матрица смежности** - это вид представления графа в виде матрицы, когда пересечение столбцов и строк задаёт дуги.

Матричные представления графа





Матрица смежности

$$\begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix}$$

$$\begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix}$$

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

```
#include <iostream>
#include <vector>
#include <fstream>
using namespace std;

class Graph {
private:
```

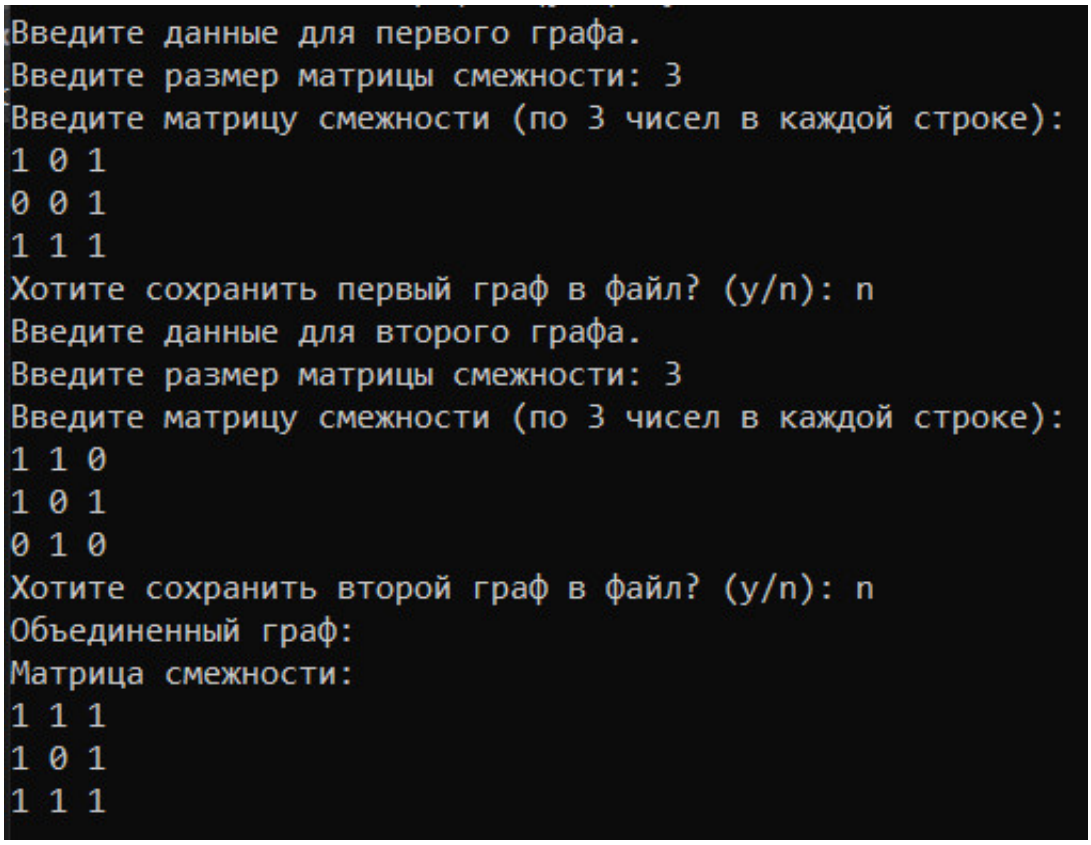
```
    vector<vector<int>> Matrix_smezh;
public:
    Graph(int razmer = 0) {
        Matrix_smezh.resize(razmer, vector<int>(razmer, 0));
    }

    void addRebro(int v1, int v2, int ves) {
        if (v1 < Matrix_smezh.size() && v2 < Matrix_smezh.size())
           {
            Matrix_smezh[v1][v2] = ves;
            Matrix_smezh[v2][v1] = ves; //

        }
    }

    void UnionGraphs(const Graph& drugoy) {
        int maxSize = max(Matrix_smezh.size(), drugoy.Matrix_smezh
           .size());
        resizeMatrix(maxSize);
        for (int i = 0; i < drugoy.Matrix_smezh.size(); ++i) {
            for (int j = 0; j < drugoy.Matrix_smezh[i].size(); ++j
               ) {
                if (drugoy.Matrix_smezh[i][j]) {
                    Matrix_smezh[i][j] = drugoy.Matrix_smezh[i][j
                       ];
                }
            }
        }
    }

    void displayGraph() const {
        cout << "Matrica␣Smejnosti:" << endl;
        for (const auto& row : Matrix_smezh) {
            for (int val : row) {
                cout << val << "␣";
            }
            cout << endl;
        }
    }

    void resizeMatrix(int newSize) {
        Matrix_smezh.resize(newSize);
        for (auto& row : Matrix_smezh) {
            row.resize(newSize, 0);
        }
    }

    const vector<vector<int>>& getMatrix_smezh() const {
        return Matrix_smezh;
    }
};
```

```
54
55  void writeGraphToFile(const Graph& graph, const string& filename)
        {
56      ofstream file(filename);
57      if (!file.is_open()) {
58          cout << "Ne␣ydalos␣otkrit␣file:␣" << filename << endl;
59          return;
60      }
61      const auto& Matrix_smezh = graph.getMatrix_smezh();
62      file << Matrix_smezh.size() << endl;
63      for (const auto& row : Matrix_smezh) {
64          for (int val : row) {
65              file << val << "␣";
66          }
67          file << endl;
68      }
69      file.close();
70      cout << "Graph␣zapisan␣v␣file␣" << filename << endl;
71  }
72
73  void readGraphFromFile(Graph& graph, const string& filename) {
74      ifstream file(filename);
75      if (!file.is_open()) {
76          cout << "Ne␣ydalos␣otkrit␣file:␣" << filename << endl;
77          return;
78      }
79      int size;
80      file >> size;
81      Graph tempGraph(size);
82      for (int i = 0; i < size; ++i) {
83          for (int j = 0; j < size; ++j) {
84              int edge;
85              file >> edge;
86              tempGraph.addRebro(i, j, edge);
87          }
88      }
89      graph.UnionGraphs(tempGraph);
90      file.close();
91  }
92
93  Graph inputGraph() {
94      int size;
95      cout << "Vvedite␣razmer␣matrici␣smejnosti:␣";
96      cin >> size;
97      Graph graph(size);
98      cout << "Vvedite␣matricy␣smejnosty␣(po␣" << size << "␣chisel␣v
            ␣kajdoi␣stroke):" << endl;
99      for (int i = 0; i < size; ++i) {
100         for (int j = 0; j < size; ++j) {
101             int rebro;
102             cin >> rebro;
```

```
103                 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



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.

Источники

- https://www.geeksforgeeks.org/what-is-unidirected-graph-undirected-graph-meaning/
- https://habr.com/ru/articles/564594/
- https://en.wikipedia.org/wiki/Graph$_theory$