

Отчёт по расчётной работе по дисциплине ПиОИВИС

Цель

Ознакомиться с понятием графов, узнать какие способы представления графов существуют, научиться решать теоретико-графовые задачи.

Задача

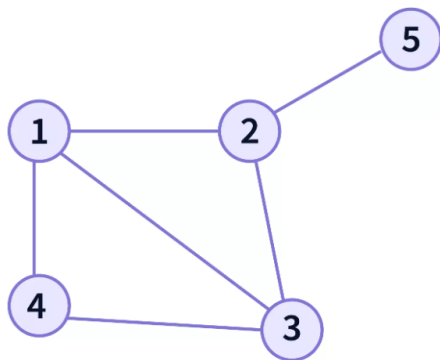
Создать алгоритм, который будет брать из файла данные и определять минимальную степень ребра в неориентированном графе.

Вариант

2.5 список смежности (список инцидентности)

Список ключевых понятий (определения)

- **Граф** – математическая абстракция реальной системы любой природы, объекты которой обладают парными связями.
- **Неориентированный граф (Undirected Graph)**: рёбра не имеют направления. Если существует ребро между А и В, то это означает, что А соединена с В и В соединена с А.



- **Инцидентность** – понятие, используемое только в отношении ребра и вершины.
- **Смежность** - понятие, используемое в отношении только двух рёбер либо только двух вершин.


```
2 3
1 4
1 5
2 3
3
```

- graph3.txt

```
1 2
1 3
2 4
3 5
4 5
5 6
6 7
7 8
```

- graph4.txt

```
1 2
1 3
2 4
2 5
3 6
4 5
5 6
6 7
7 8
```

Алгоритм

1. Выбрать файл с готовым графом и открыть его.
2. Определить максимальную вершину.
3. Проверить корректность графа, а именно все соседи вершины должны быть в пределах допустимого диапазона (индексы от 0 до $N - 1$) и количество вершин в графе должно совпадать с номером максимальной вершины.
4. Вывести список смежности.
5. Подсчитать для каждой вершины степень (количество соседей).
6. Вывести минимальную степень.
7. Завершить программу.

Код

```
1  #include <iostream>
2  #include <fstream>
3  #include <vector>
4  #include <string>
5
6  using namespace std;
7
8  int main() {
9      string nameOfFile;
10     int numberOfGraph;
11     cout << "Enter the file number where the graph is stored: ";
12     cin >> numberOfGraph;
13
14     switch (numberOfGraph) {
15     case 1: nameOfFile = "graph.txt"; break;
16     case 2: nameOfFile = "graph1.txt"; break;
17     case 3: nameOfFile = "graph2.txt"; break;
18     case 4: nameOfFile = "graph3.txt"; break;
19     case 5: nameOfFile = "graph4.txt"; break;
20     default:
21         cout << "Invalid file number." << endl;
22         return 0;
23     }
24
25     ifstream fin(nameOfFile);
26     if (!fin.is_open()) {
27         cout << "The file could not be opened!" << endl;
28         return 0;
29     }
30
31     vector<vector<int>> adjacencyList;
32     string line;
33     int maxVertex = 0;
34
35
36     while (getline(fin, line)) {
37         vector<int> neighbors;
38         int vertex;
39         size_t pos = 0;
40
41         while (pos < line.size()) {
42             if (isdigit(line[pos])) {
43                 size_t end;
44                 vertex = stoi(line.substr(pos), &end);
45                 neighbors.push_back(vertex - 1);
46                 maxVertex = max(maxVertex, vertex);
47                 pos += end;
48             }
```

```

49         else {
50             pos++;
51         }
52     }
53     adjacencyList.push_back(neighbors);
54 }
55
56 fin.close();
57
58
59 bool isAdjacencyList = true;
60 for (size_t i = 0; i < adjacencyList.size(); i++) {
61     for (size_t j = 0; j < adjacencyList[i].size(); j++) {
62         if (adjacencyList[i][j] < 0 || adjacencyList[i]
63             ][j] >= adjacencyList.size()) {
64             isAdjacencyList = false;
65             break;
66         }
67     }
68     if (!isAdjacencyList) break;
69 }
70
71 if (!isAdjacencyList) {
72     cout << "The graph is not a valid adjacency list (incorrect
73         neighbor)." << endl;
74     return 1;
75 }
76
77 if (adjacencyList.size() != maxVertex) {
78     cout << "The graph is not correct. Some vertices are missing
79         from the adjacency list." << endl;
80     return 1;
81 }
82
83 cout << "Read the adjacency list:" << endl;
84 for (size_t i = 0; i < adjacencyList.size(); i++) {
85     cout << "Vertex " << i + 1 << ": ";
86     for (size_t j = 0; j < adjacencyList[i].size(); j++) {
87         cout << adjacencyList[i][j] + 1 << " ";
88     }
89     cout << endl;
90 }
91
92 if (adjacencyList.empty()) {
93     cout << "The count is empty!" << endl;
94     return 1;
95 }

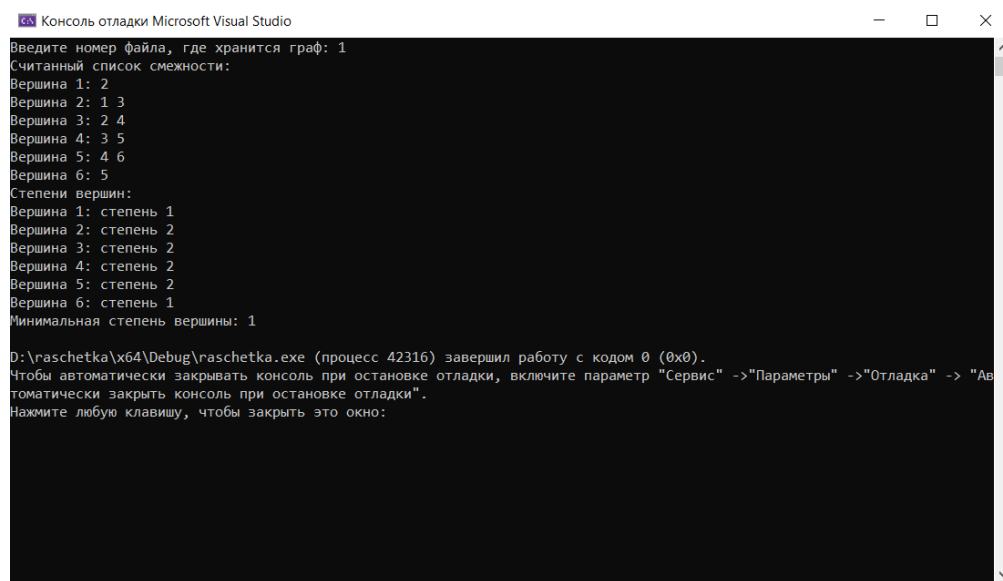
```

```

96     int minDegree = adjacencyList[0].size();
97     cout << "Degrees of vertices:" << endl;
98     for (size_t i = 0; i < adjacencyList.size(); i++) {
99         int degree = adjacencyList[i].size();
100        cout << "Vertex" << i + 1 << ": degree" << degree << endl;
101        if (degree < minDegree) {
102            minDegree = degree;
103        }
104    }
105
106    cout << "The minimum degree of the vertex:" << minDegree << endl;
107
108    return 0;
109 }

```

Пример выполнения работы



```

Консоль отладки Microsoft Visual Studio
Введите номер файла, где хранится граф: 1
Считанный список смежности:
Вершина 1: 2
Вершина 2: 1 3
Вершина 3: 2 4
Вершина 4: 3 5
Вершина 5: 4 6
Вершина 6: 5
Степени вершин:
Вершина 1: степень 1
Вершина 2: степень 2
Вершина 3: степень 2
Вершина 4: степень 2
Вершина 5: степень 2
Вершина 6: степень 1
Минимальная степень вершины: 1
D:\raschetka\x64\Debug\raschetka.exe (процесс 42316) завершил работу с кодом 0 (0x0).
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
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.