Отчёт по расчётной работе по дисциплине ПиОИВиС

Барчук Алина Эдуардовна

20 декабря 2024 г.

Тема: Графы

Цель:

Найти минимальное и среднее расстояние между периферийными вершинами неориентированного графа

Задача:

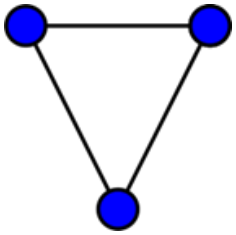
создать программу, котрая будет находить минимальное и среднее расстояние между периферийными вершинами неориенти- рованного графа

Вариант:

2.10(ми)

Список ключевых понятий:

- **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.



- **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).

V	1-2	1-3	2-4	2-5	3-5
1	1	1	0	0	0
2	1	0	1	1	0
3	0	1	0	0	1
4	0	0	1	0	0
5	0	0	0	1	1

- **Периферийные вершины (или периферийные узлы) в графах** — это вершины, которые имеют степень, равную 1. Это означает, что каждая такая вершина соединена только с одной другой вершиной через одно ребро.
- Расстояние между периферийными вершинами в графе называется **дистанцией или расстоянием**. Это количество рёбер (или шагов), которые необходимо пройти, чтобы добраться от одной периферийной вершины до другой.

Файлы с содержанием матрицы инцидентности:

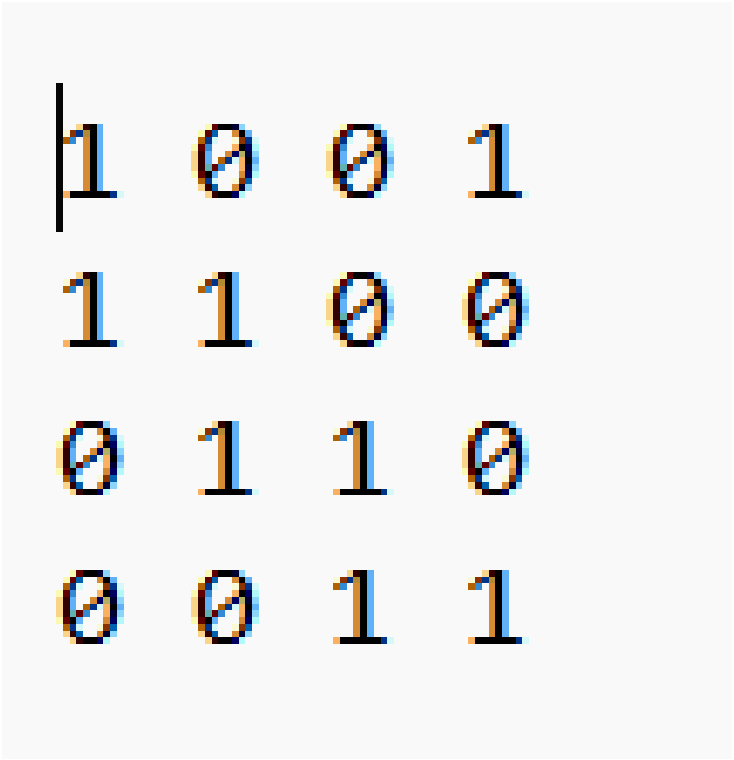
- `graph.txt`

```
1 0 0
1 1 0
0 1 1
0 0 1
```

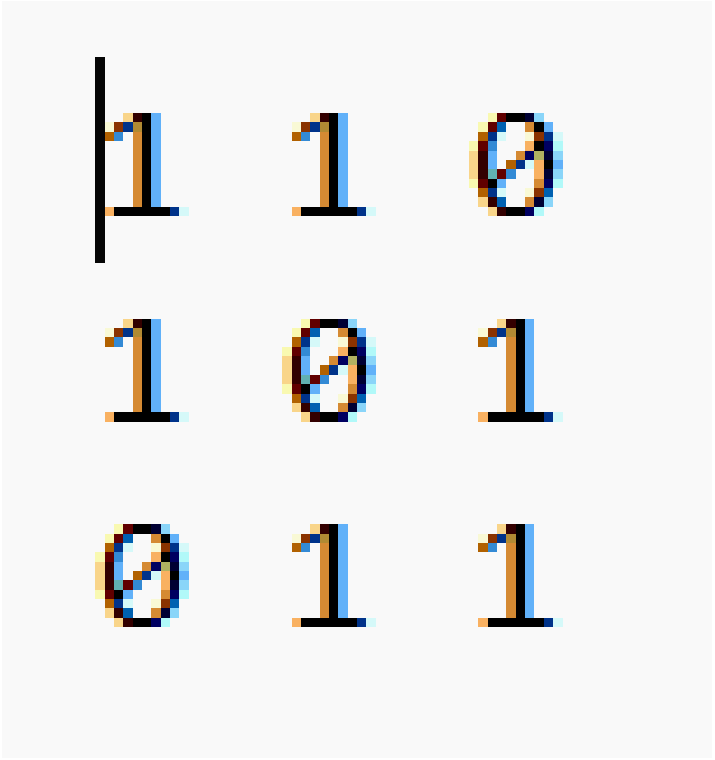
- graph2.txt

```
1 0 0 0
0 1 0 0
0 0 1 0
1 1 0 1
0 0 1 1
```

- graph3.txt



- graph4.txt



- graph5.txt

```
1 0 0 0
1 1 0 0
0 1 0 0
0 0 1 0
0 0 1 1
0 0 0 1
```

Алгоритм

1. Выбрать файл с матрицей инцидентности и открыть его.
2. Проверка и вывод матрицы.
3. Посчитать и выявить периферийные вершины.
4. Вычисление расстояний между периферийными вершинами.
5. Предоставление информации о минимальном и среднем расстоянии.

Код:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

bool isIncidenceMatrix(const vector<vector<int>>& matrix) {
    int edges = matrix[0].size();
    for (const auto& row : matrix) {
```

```
        if (row.size() != edges) {
            return false;
        }
    }

    for (int j = 0; j < edges; ++j) {
        int count = 0;
        for (int i = 0; i < matrix.size(); ++i) {
            count += abs(matrix[i][j]);
        }
        if (count != 2) {
            return false;
        }
    }
    return true;
}


vector<int> bfs(const vector<vector<int>>& matrix, int start) {
    int vertices = matrix.size();
    vector<int> distances(vertices, -1);

    vector<int> queue;
    queue.push_back(start);
    distances[start] = 0;

    int front = 0;
    while (front < queue.size()) {
        int current = queue[front++];

        for (int j = 0; j < matrix[0].size(); ++j) {
            if (matrix[current][j] == 1) {
                for (int i = 0; i < vertices; ++i) {
                    if (i != current && matrix[i][j] == 1 && distances[i] == -1) {
                        distances[i] = distances[current] + 1;
                        queue.push_back(i);
                    }
                }
            }
        }
    }

    return distances;
}

int main() {
    setlocale(LC_ALL, "rus");
    string filename;
```

```
int fileChoice;

cout << "Выберите файл для загрузки графа (1-5):" << endl;
cin >> fileChoice;

switch (fileChoice) {
case 1:
    filename = "graph.txt";
    break;
case 2:
    filename = "graph2.txt";
    break;
case 3:
    filename = "graph3.txt";
    break;
case 4:
    filename = "graph4.txt";
    break;
case 5:
    filename = "graph5.txt";
    break;
default:
    cout << "Неверный выбор файла!" << endl;
    return 1;
}

ifstream file(filename);
if (!file.is_open()) {
    cout << "Не удалось открыть файл!" << endl;
    return 1;
}

vector<vector<int>> matrix;

string line;
while (getline(file, line)) {
    vector<int> row;
    size_t pos = 0;
    while ((pos = line.find(' ')) != string::npos) {
        row.push_back(stoi(line.substr(0, pos)));
        line.erase(0, pos + 1);
    }
    row.push_back(stoi(line));
    matrix.push_back(row);
}

file.close();
```

```
if (!isIncidenceMatrix(matrix)) {
    cout << "Это не матрица инцидентности!" << endl;
    return 1;
}

int rowCount = matrix.size();
int colCount = matrix[0].size();
cout << "Матрица инцидентности:" << endl;
for (int i = 0; i < rowCount; i++) {
    for (int j = 0; j < colCount; j++) {
        cout << matrix[i][j] << " ";
    }
    cout << endl;
}

int vertices = matrix.size();
vector<int> peripheralVertices;

for (int i = 0; i < vertices; ++i) {
    int degree = 0;
    for (int j = 0; j < matrix[0].size(); ++j) {
        if (matrix[i][j] == 1) {
            degree++;
        }
    }
    if (degree == 1) {
        peripheralVertices.push_back(i);
    }
}

if (peripheralVertices.empty()) {
    cout << "Нет периферийных вершин." << endl;
}
else {
    cout << "Периферийные вершины: ";
    for (int vertex : peripheralVertices) {
        cout << vertex << " ";
    }
    cout << endl;

    vector<int> distances;

    for (int i = 0; i < peripheralVertices.size(); ++i) {
        vector<int> bfsDistances = bfs(matrix, peripheralVertices[i]);
        for (int j = i + 1; j < peripheralVertices.size(); ++j) {
            int dist = bfsDistances[peripheralVertices[j]];
            distances.push_back(dist);
            cout << "Расстояние от вершины " << peripheralVertices[i] <<
```

```
" до вершины " << peripheralVertices[j] << " равно: " << dist << endl;
            }
        }

        if (!distances.empty()) {
            int minDistance = distances[0];
            for (int dist : distances) {
                if (dist < minDistance) {
                    minDistance = dist;
                }
            }

            double sum = 0;
            for (int dist : distances) {
                sum += dist;
            }
            double avgDistance = sum / distances.size();

            cout << "Минимальное расстояние: " << minDistance << endl;
            cout << "Среднее расстояние: " << avgDistance << endl;
        }
        else {
            cout << "Нет расстояний для вычисления." << endl;
        }
    }

    return 0;
}
```

Пример работы кода

```
Выберите файл для загрузки графа (1-5):
4
Матрица инцидентности:
1 1 0
1 0 1
0 1 1
Нет периферийных вершин.
```

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.