Rzeszów, 15.01.2022

PROJEKT NR III

ALGORYTMY I STRUKTURY DANYCH

Jakub Misiło
Inżynieria i Analiza Danych I rok, grupa V

1. **Temat Projektu**

Napisz program, który dla zadanego grafu skierowanego reprezentowanego przy pomocy listy krawędzi zapisanych przy pomocy dwóch tablic wyznaczy i wypisze następujące informacje:

a) Wszystkich sąsiadów dla każdego wierzchołka grafu
b) Wszystkie wierzchołki, które są sąsiadami każdego wierzchołka
c) Stopnie wchodzące wszystkich wierzchołków
d) Stopnie wychodzące wszystkich wierzchołków
e) Wszystkie wierzchołki izolowane
f) Wszystkie pętle
g) Wszystkie krawędzie dwukierunkowe

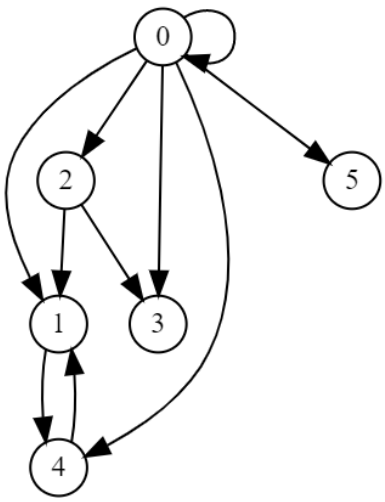
2. **Cel projektu**

Celem projektu było poznanie i dokonanie właściwej implementacji struktury danych typu graf skierowany reprezentowany przy pomocy listy krawędzi.

3. **Podstawy teoretyczne**

Graf skierowany - definiuje się jako uporządkowaną parę zbiorów. Pierwszy z nich zawiera wierzchołki grafu, a drugi składa się z krawędzi grafu, czyli uporządkowanych par wierzchołków. Ruch po grafie możliwy jest tylko w kierunkach wskazywanych przez krawędzie. Graf skierowany można sobie wyobrazić jako sieć ulic, z których każda jest jednokierunkowa. Ruch pod prąd jest zakazany. Najczęściej grafy skierowane przedstawia się jako zbiór punktów reprezentujących wierzchołki połączonych strzałkami (stąd nazwa) albo łukami zakończonymi grotem (strzałką, zwrotem).
Stopień wchodzący – liczba krawędzi wchodzących do wierzchołka.
Stopień wychodzący – liczba krawędzi wychodzących z wierzchołka.
Pętla - sytuacja w której wierzchołek łączy się z sobą samym.

2. **Opis zaimplementowanego grafu**



Schemat grafu, który został zaimplementowany w celu sprawdzenia jego parametrów

3. **Kod zaimplementowanej biblioteki**

```
#include <iostream>
#include <vector>
#include <set>

using namespace std;

struct edge {
    int ver_out, ver_in; // wierzcholek wychodzacy, wierzcholek wchodzacy
};

class DiGraph {

    public:
            vector<edge> edges;
            vector<int> vertices;

            int n, N;

            DiGraph(vector<edge> edges, vector<int> vertices) {
                this->edges = edges;
                this->vertices = vertices;
                this->N = this->edges.size();
                this->n = this->vertices.size();
            }

            // 1
            void displayAdj() {
                for (int i=0; i<n; i++) {
                    vector<int> adj = {};

                    for (int j=0; j<N; j++) {

                        if (this->vertices[i] == this->edges[j].ver_out) adj.push_back(edges[j].ver_in);
                    }

                    cout << "Sasiedzi wierzcholka " << this->vertices[i] << ": ";
                    for (int j=0; j<adj.size(); j++ ) cout << adj[j] << ", ";
                    cout << endl;
                }
            }

            // 2
            void everyVerAdj() {
                set<int> vert = {};

                for (int i = 0; i < this->n; i++) {
                    set<int> adj = {};

                    for (edge j: this->edges) {
                        if (j.ver_in == this->vertices[i]) adj.insert(this->vertices[i]);
                    }

                    if(adj.size() >= this->n) vert.insert(this->vertices[i]);
                }

                cout << "Wierzcholki bedace sasiadami kazdego wierzcholka:" << endl;
                for(int i: vert) cout << i << ", ";
            }
```

```
// 3
void degOut() {
    for (int i=0; i<n; i++) {
        int degOut = 0;

        for (int j=0; j<N; j++)
            if (this->vertices[i] == this->edges[j].ver_out) degOut++;

        cout << "Stopien wychodzacy wierzcholka " << this->vertices[i] << ": " << degOut << endl;
    }
}

// 4
void degIn() {
    for(int i=0; i<n; i++) {
        int degIn = 0;

        for (int j=0; j<N; j++)
            if (this->vertices[i] == this->edges[j].ver_in) degIn++;

        cout << "Stopien wchodzacy wierzcholka " << this->vertices[i] << ": " << degIn << endl;
    }
}



// 5
void sepVertices() {
    vector<int> sepVert = {};

    for(int i: this->vertices) {
        int counter = 0;
        for(edge j: this->edges) {
            if(i == j.ver_out) counter++;
        }

        if(!counter) sepVert.push_back(i);
    }

    cout << "Wierzcholki izolowane: ";
    for (int i: sepVert) cout << i << ", ";
    cout << endl;
}

// 6
void disLoops() {
    vector<int> loops = {};

    for (edge i: this->edges)
        if(i.ver_out == i.ver_in)
            loops.push_back(i.ver_out);

    cout << "Petle:" << endl;
    for (int i: loops) cout << "(" << i << ", " << i << ")" << endl;

}
```

```
    // 7
    void twoDirEdges() {
        vector<edge> twoDir = {};

        for (edge i: this->edges) {
            for (edge j: this->edges) {
                if(i.ver_in == j.ver_out && i.ver_out == j.ver_in && i.ver_in != i.ver_out) {
                    vector<int> helper = {j.ver_in, j.ver_out};

                    // dodajemy krawedz tylko jesli krawedz odwrotna nie wystapila w twoDir
                    if (twoDir.size()) {
                        for (edge z: twoDir) {
                            if (z.ver_out != helper[0] && z.ver_in != helper[1])
                                twoDir.push_back(j);
                        }
                    } else twoDir.push_back(j);
                }
            }
        }

        cout << "Krawedzie dwukierunkowe: " << endl;
        for (edge i: twoDir) cout << "{" << i.ver_out << ", " << i.ver_in << "}" << endl;
    }
};
```

4. Rezultaty

1. **Wszyscy sąsiedzi dla każdego wierzchołka grafu:**

- Sasiedzi wierzcholka 0:

0, 1, 2, 3, 4, 5

- Sasiedzi wierzcholka 1:

4,

- Sasiedzi wierzcholka 2:

1, 3

- Sasiedzi wierzcholka 3:
- Sasiedzi wierzcholka 4:

1,

- Sasiedzi wierzcholka 5:

2. **Wszystkie wierzchołki będące sąsiadami każdego wierzchołka**

- Wierzcholki bedace sasiadami kazdego innego wierzcholka:

3. **Stopnie wychodzące wszystkich wierzchołków**

- Stopien wychodzacy wierzcholka 0: 6
- Stopien wychodzacy wierzcholka 1: 1
- Stopien wychodzacy wierzcholka 2: 2
- Stopien wychodzacy wierzcholka 3: 0
- Stopien wychodzacy wierzcholka 4: 1
- Stopien wychodzacy wierzcholka 5: 0

4. **Stopnie wchodzące wszystkich wierzchołków**

- Stopien wchodzacy wierzcholka 0: 1
- Stopien wchodzacy wierzcholka 1: 3
- Stopien wchodzacy wierzcholka 2: 1
- Stopien wchodzacy wierzcholka 3: 2
- Stopien wchodzacy wierzcholka 4: 2
- Stopien wchodzacy wierzcholka 5: 1

5. **Wszystkie wierzchołki izolowane**
   - Wierzcholki izolowane grafu:

3, 5

6. **Wszystkie pętle**

(0, 0)

7. **Wszystkie krawędzie dwukierunkowe**
   - Krawedzie dwukierunkowe:

{1, 4}

*** Link do repozytorium**