МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
Київський національний університет імені Тараса Шевченка
Кафедра програмних систем і технологій

Звіт з лабораторної роботи 2.7

Тема: «Алгоритми на неорієнтованих графах»

Виконав: студент групи ІПЗ - 12
Богатько Олександр
Геннадійович
Перевірила: викладач
Юрчук Ірина Аркадіївна

Київ 2020

1.Алгоритм Крускала(Побудови остовного дерева мінімальної вартості).

Код програми:

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

struct Set
{ public:
    vector<int> set;
    Set(int k)
    {
        set.push_back(k); }
};
struct Edge
{
public:
    int v1; int v2; int weight; Edge(int f,int s, int w) {
        v1 = f;
        v2 = s;
        weight = w;
    }
    };
struct DisjointSets
{
public:
    int NumberSets = 0; vector <Set> components; DisjointSets(int num)
    {
        NumberSets = num;
        for(int i = 0; i < num; ++i)
            components.push_back(Set(i));
    }
    int Find(int vertex)
    {
        for(int i = 0; i < NumberSets; ++i)
        {
            for(int j = 0; j <components[i].set.size(); ++j)
                if (vertex == components[i].set[j]) return i; }
    } };
void Print(vector<Edge> res)
{
    cout << " U V Weight" << endl; for(Edge i : res)
        cout<<""<<i.v1+1<<" "<<i.v2+1<<" "<<i.weight<<endl;
}
vector<Edge> Kruskal(vector<Edge> graph, int size) {
    DisjointSets f(size);
    vector<Edge> res;
    for(int i = 0; i < graph.size(); ++i)

    {
        for(int j = 0; j < graph.size(); ++j)
            if(graph[i].weight < graph[j].weight) swap(graph[i], graph[j]);
    }
    int c = 1;
    for(Edge r : graph){
        int res1 = f.Find(r.v1); int res2 = f.Find(r.v2); if( res1 != res2){
            res.emplace_back(r);
            for(int i = 0; i < f.components[res2].set.size(); ++i)
                f.components[res1].set.emplace_back(f.components[res2].set[i]);
            f.components[res2].set.clear();
            cout << "Step " << c << ":" << endl;
            Print(res);
            c++; }
    }
    return res; }
int main() {
    int vert, v1, v2, weight, edge;
    cout << "Enter number of vertex and edges: ";
    cin >> vert >> edge;
    cout << "Enter first vertex, second vertex and weight of the edge: " << endl;
    vector<Edge> graph;
    for(int i = 0; i < edge; ++i){
        cin >> v1 >> v2 >> weight;
        graph.emplace_back(v1 - 1, v2 - 1, weight); }
```

```
    vector<Edge> res;
    res = Kruskal(graph, vert);
    cout << "Kruskal's MST: " << endl;
    Print(res);
    return 0;
```

Результати роботи програми:

```
Enter number of vertex and edges: 3 6
Enter first vertex, second vertex and weight of the edge:
3 2 5
4 7 1
8 5 3
4 2 1
7 6 8
4 3 1
Step 1:
 U V Weight
4 2 1
Step 2:
 U V Weight
4 2 1
4 3 1
Kruskal's MST:
 U V Weight
4 2 1
4 3 1

Process finished with exit code 0
```

```
Enter number of vertex and edges: 6 8
Enter first vertex, second vertex and weight of the edge:
3 6 8
4 2 8
1 6 3
9 1 3
5 2 1
2 7 4
9 7 2
2 9 5
Step 1:
 U V Weight
5 2 1
Step 2:
 U V Weight
5 2 1
1 6 3
Step 3:
 U V Weight
5 2 1
1 6 3
9 1 3
Step 4:
 U V Weight
5 2 1
1 6 3
9 1 3
4 2 8
Kruskal's MST:
 U V Weight
5 2 1
1 6 3
9 1 3
4 2 8

Process finished with exit code 0
```

2.Алгоритм Прима(Побудови остовного дерева мінімальної вартості).

Код програми:

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

const int MAXN = 100, INF = 10000000;

struct Edge
{
public:
    int v1; int v2; int weight; Edge(int a,int b, int c){
        v1 = a;
        v2 = b; weight = c;
    }
};
int n, u, v;
vector <int> included, g[MAXN], w[MAXN]; vector <bool> used(MAXN, false);
vector <Edge> res;
void Print(vector<Edge> res)
{
    cout << " U V Weight" << endl;
    for(Edge i : res)
        cout<<""<<i.v1+1<<" " <<i.v2+1<<" "<<i.weight<<endl; }
void Prim() {
    int c = 0;
    used[0] = true; included.push_back(0);
    for (int i = 1; i < n; i++) {
        int minWeight = INF, minTop;
        for (int j = 0; j < included.size(); j++) {
            int cur = included[j];
            for (int k = 0; k < g[cur].size(); k++) {
                int next = g[cur][k];
                if (!used[next] && w[cur][k] < minWeight)
                {
                    minWeight = w[cur][k]; u = cur;
                    v = next;
                    minTop = next;
                }
            }
        }
        res.emplace_back(u, v, minWeight); used[minTop] = true; included.push_back(minTop);
        cout << "Step " << c << ":" << endl; Print(res);
        c++;
    }
}
int main(){
    int m, v1, v2,
            weight;

    cout << "Enter number of vertices and edges: ";
    cin >> n >> m;
    cout << "Enter first vertex, second vertex and edge weight:" << endl;
    for (int i = 0; i < m; i++){
     cin >> v1 >> v2 >> weight;
        v1--;
        v2--;
        g[v1].push_back(v2);
        w[v1].push_back(weight);
        g[v2].push_back(v1);
        w[v2].push_back(weight);
    }
    Prim();
    cout << "Prim's MST:" << endl; Print(res);
}
```

Результати роботи програми:

```
/Users/aleksandrbogatko/Desktop/Alexandr_Bogatko_I
Enter number of vertices and edges: 6 9
Enter first vertex, second vertex and edge weight:
8 2 5
4 8 1
3 8 5
9 1 4
3 8 5
1 3 2
9 7 8
8 6 7
6 4 5
Step 0:
 U V Weight
1 3 2
Step 1:
 U V Weight
1 3 2
1 9 4
Step 2:
 U V Weight
1 3 2
1 9 4
3 8 5
Step 3:
 U V Weight
1 3 2
1 9 4
3 8 5
8 4 1
Step 4:
 U V Weight
1 3 2
1 9 4
3 8 5
8 4 1
8 2 5
Prim's MST:
 U V Weight
1 3 2
1 9 4
3 8 5
8 4 1
8 2 5

Process finished with exit code 0
```

```
Enter number of vertices and edges: 5 9
Enter first vertex, second vertex and edge weight:
2 5 9
4 1 8
3 5 7
8 2 5
9 1 2
3 5 9
8 2 5
4 3 9
2 1 3
Step 0:
 U V Weight
1 9 2
Step 1:
 U V Weight
1 9 2
1 2 3
Step 2:
 U V Weight
1 9 2
1 2 3
2 8 5
Step 3:
 U V Weight
1 9 2
1 2 3
2 8 5
1 4 8
Prim's MST:
 U V Weight
1 9 2
1 2 3
2 8 5
1 4 8

Process finished with exit code 0
```