Assignment #3

1. Kruskal's Algorithm & Prim's Algorithm

Date of Performance: 20-05-2024

Date of Submission: 27-05-2024

Student ID: 20220104147

Name: Md. Redwan Hossen

Group: C_2

```cpp
#include<bits/stdc++.h>
using namespace std;


int findSet(int x, vector<int> &parent)
{
    if (parent[x] != x)
    {
        parent[x] = findSet(parent[x], parent);
    }
    return parent[x];
}



void unionSets(int x, int y, vector<int> &parent, vector<int> &rank)
{
    int rootX = findSet(x, parent);
    int rootY = findSet(y, parent);

    if (rootX != rootY)
    {
        if (rank[rootX] > rank[rootY])
        {
            parent[rootY] = rootX;
        }
        else if (rank[rootX] < rank[rootY])
        {
            parent[rootX] = rootY;
        }
        else
        {
            parent[rootY] = rootX;
            rank[rootX]++;
        }
    }
}



bool sortByWeight(const pair<pair<int, int>, int> &a, const pair<pair<int, int>, int> &b)
{
    return a.second < b.second;
}



pair<vector<pair<int, int>>, int> kruskal(int V, vector<pair<pair<int,
int>, int>> &edges)
{
    vector<pair<int, int>> mst;
    vector<int> parent(V), rank(V, 0);
    int totalCost = 0;


    for (int i = 0; i < V; ++i)
    {
        parent[i] = i;
```

```
    }

    sort(edges.begin(), edges.end(), sortByWeight);

    for (int i = 0; i < edges.size(); ++i)
    {
        int u = edges[i].first.first;
        int v = edges[i].first.second;
        int weight = edges[i].second;

        if (findSet(u, parent) != findSet(v, parent))
        {
            mst.push_back({u, v});
            totalCost += weight;
            unionSets(u, v, parent, rank);
        }
    }

    return {mst, totalCost};
}

int main()
{
    int V, E;
    cout << "Enter the number of vertices: ";
    cin >> V;
    cout << "Enter the number of edges: ";
    cin >> E;

    vector<pair<pair<int, int>, int>> edges(E);

    cout << "Enter the edges with their weights (u v w):" << endl;
    for (int i = 0; i < E; i++)
    {
        cin >> edges[i].first.first >> edges[i].first.second >> edges[i].second;
    }

    auto result = kruskal(V, edges);
    vector<pair<int, int>> mst = result.first;
    int totalCost = result.second;

    cout << "T={";
    int mstIndex = 0;
    for (int i = 0; i < edges.size(); ++i)
    {

      if (mstIndex < mst.size() && edges[i].first.first ==
      mst[mstIndex].first && edges[i].first.second == mst[mstIndex].second)
        {
```

```
            cout << "(" << edges[i].first.first << "," << edges[i].first.second << ")";
            ++mstIndex;
        }
        else
        {
           cout << "(" << edges[i].first.first << "," << edges[i].first.second << ")x";
        }
        if (i != edges.size() - 1) cout << ",";
    }
    cout << "} ; here (a,b)x means this pair is not allowed to sit here" << endl;

    cout << "Cost= " << totalCost << endl;
    cout << "Total edges= " << mst.size() << endl;

    return 0;
}
```

```
/*

Enter the number of vertices: 7
Enter the number of edges: 11
Enter the edges with their weights (u v w):
0 1 2
0 5 14
0 6 8
6 5 21
1 5 25
4 5 13
2 5 17
1 2 19
2 3 9
2 4 5
3 4 1
T={(3,4),(0,1),(2,4),(0,6),(2,3)x,(4,5),(0,5),(2,5)x,(1,2)x,(6,5)x,(1,5)x};
here (a,b)x means this pair is not allowed to sit here
Cost= 43
Total edges= 6

*/
```