RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

SESSIONAL TASK-09

COURSE NAME: SESSIONAL BASED ON CSE-2201
COURSE CODE: CSE-2102

SUBMITTED TO-

BIPRODIP PAL
Assistant Professor
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

SUBMITTED BY-

SRABONTI DEB
Roll- 1803163
Section-C
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

SUBMISSION DATE – 15 AUGUST, 2021

Problem Statement: Generate random integers V (total vertices) and E (total edges) followed by E edges (u,v) and corresponding weights for an undirected graph.

Code:(Using path compression)

```
#include <bits/stdc++.h>
using namespace std;
const int mx = 100000;
pair <long long, pair<int, int> > store[mx];
int indx[mx], vertex, path;

void init()
{
    for(int i = 0;i < mx;++i)
        indx[i] = i;
}

int pick_root(int t)
{
    while(indx[t] != t)
    {
        indx[t] = indx[indx[t]];
        t = indx[t];
    }
    return t;
}

void union_func(int a, int b)
{
    int store = pick_root(a);
    int q = pick_root(b);
    indx[store] = indx[q];
}

long long kruskal(pair<long long, pair<int, int> > store[])
{
    int a, b;
    long long cost, min_cost = 0;
    for(int i = 0;i < path;++i)
    {
```

```
        a = store[i].second.first;
        b = store[i].second.second;
        cost = store[i].first;
        if(pick_root(a) != pick_root(b))
        {
            min_cost += cost;
            union_func(a, b);
        }
    }
    return min_cost;
}
int main()
{
    int v, e;
    long long w, cost, min_cost;
    init();
    cin >> vertex >> path;
    for(int i = 0;i < path;++i)
    {
        cin >> v >> e >> w;
        store[i] = make_pair(w, make_pair(v, e));
    }
    sort(store, store + path);

    auto d1 = chrono::steady_clock::now();
    min_cost = kruskal(store);
    auto d2 = chrono::steady_clock::now();

    double difference = double(chrono::duration_cast <chrono::nanoseconds> (d2-d1).count());

    cout <<"Total Cost: "<< min_cost << endl;
    cout<<"The time Required for kruskal(path compression): "<<difference/1000000<<" millisecond";
    return 0;
}

```

Output:

```
"E:\CSE 2-2\class video 2-2\computer algorithm\Lab reports\Lab-9\Untitled1.exe"
3 2
1 2 5
3 2 8
Total Cost: 13
The time Required for kruskal(path compression): 0.0002 millisecond
Process returned 0 (0x0)   execution time : 1.692 s
Press any key to continue.
```

Code:(Without path compression)

```
#include <bits/stdc++.h>
using namespace std;
int find(int i,vector<int>&parent)
{
    while (parent[i] != i)
        i = parent[i];
    return i;
}

int main()
{
    int V,E;
    cin>>V>>E;
    int i,j;
    long long c[100][100]={INT_MAX};
    for(i=0;i<=V+1;i++)
    {
        for(j=0;j<=V+1;j++)
        {
            c[i][j]=INT_MAX;
        }
    }
```

```
    for(int i = 0; i<E; i++)
    {
        int u, v, w;
        cin >> u >> v >> w;
        c[u][v]=w;
        c[v][u]=w;

    }
    auto d1 = chrono::steady_clock::now();
    vector<int> parent(V);
    for(int i = 0; i<V; i++)
        parent[i] = i;
    int min_cost = 0;

    int edge_count = 0;
    while (edge_count < E)
    {
        int min = INT_MAX, a = -1, b = -1;
        for (int i = 0; i <=V; i++)
        {
            for (int j = 0; j <=V; j++)
            {
                if (find(i,parent) != find(j,parent) && c[i][j] < min)
                {
                    min = c[i][j];
                    a = i;
                    b = j;
                }
            }
        }
        int aa = find(a,parent);
        int bb = find(b,parent);
        parent[a] = b;
        edge_count++;
        min_cost += min;
    }
     auto d2 = chrono::steady_clock::now();
    double difference = double(chrono::duration_cast <chrono::nanoseconds> (d2-d1).count());
```

```
    cout<<"Total cost: "<<min_cost<<endl;
    cout<<"The time Required for kruskal(without path compression): "<<difference/1000000<<" millisecond";
    return 0;
}
```

Output:

```
"E:\CSE 2-2\class video 2-2\computer algorithm\Lab reports\Lab-9\Untitled1.exe"
3 2
1 2 5
3 2 8
Total cost: 13
The time Required for kruskal(path compression): 0.0021 millisecond
Process returned 0 (0x0)   execution time : 13.220 s
Press any key to continue.
```

V	E	Time (Path compression/Rank based approach)	Time(Without path compression/rank approach)	Solution (Total cost)
3	2	0.0002	0.0021	13
5	7	0.0005	0.009	9
9	14	0.0007	0.0277	37

Problem Statement: Find the Longest Common Subsequence (LCS) between two strings using dynamic programming.

Code:

```
#include<bits/stdc++.h>
using namespace std;
int Lcs(string x,string y)
{
    int m,n,i,j;
    m=x.length();
    n=y.length();
    int f_arr[m+1][n+1];
    for(i=0; i<=m; i++)
    {
        f_arr[0][i]=0;
    }
    for(i=0; i<=n; i++)
    {
        f_arr[i][0]=0;
    }
    for(i=1; i<=m; i++)
    {
        for(j=1; j<=n; j++)
        {
            if(x[i-1]==y[j-1])
            {
                f_arr[i][j]=f_arr[i-1][j-1]+1;
            }
            else
                f_arr[i][j]=max(f_arr[i][j-1],f_arr[i-1][j]);
        }
    }
    return f_arr[m][n];
}
int main()
{
  string a,b;
  cin>>a>>b;
```

```
cout<<Lcs(a,b);
}
```

Output:

```
"E:\2-2\computer algorithm\Lab-9\longest_common_subsequence.exe"
ABCDGH
AEDFHR
3
Process returned 0 (0x0)   execution time : 17.734 s
Press any key to continue.
```