CODE:-

#include <iostream>

#include <cstring>

#include <cstdlib>

using namespace std;

void lcs(char \*X, char \*Y, int m, int n)

{

int L[m + 1][n + 1];

for (int i = 0; i <= m; i++)

{

for (int j = 0; j <= n; j++)

{

if (i == 0 || j == 0)

L[i][j] = 0;

else if (X[i - 1] == Y[j - 1])

L[i][j] = L[i - 1][j - 1] + 1;

else

L[i][j] = max(L[i - 1][j], L[i][j - 1]);

}

}

int index = L[m][n];

char lcs[index + 1];

lcs[index] = '\0';

while (i > 0 && j > 0)

{

if (X[i - 1] == Y[j - 1])

{

lcs[index - 1] = X[i - 1]; // Put current character in result

i--;

j--;

index--; // reduce values of i, j and index

}

else if (L[i - 1][j] > L[i][j - 1])

i--;

else

j--;

}

cout << "LCS of " << X << " and " << Y << " is " << lcs;

}

int main()

{

char X[100];

cin >> X;

char Y[100];

cin >> Y;

int m = strlen(X);

int n = strlen(Y);

lcs(X, Y, m, n);

return 0;

}

#include <bits/stdc++.h>

using namespace std;

class DSU {

int\* parent;

int\* rank;

public:DSU(int n)

{parent = new int[n];

rank = new int[n];

for (int i = 0; i < n; i++) {

parent[i] = -1;

rank[i] = 1;

}

}

int find(int i)

{

if (parent[i] == -1)

return i;

return parent[i] = find(parent[i]);

}

void unite(int x, int y)

{

int s1 = find(x);

int s2 = find(y);

if (s1 != s2) {

if (rank[s1] < rank[s2]) {

parent[s1] = s2;

rank[s2] += rank[s1];

}

else {

parent[s2] = s1;

rank[s1] += rank[s2];

}

}

}

};lass Graph {

vector<vector<int> > edgelist;

int V;

public:

Graph(int V) { this->V = V; }void addEdge(int x, int y, int w)

{

edgelist.push\_back({ w, x, y });

}void kruskals\_mst()

{sort(edgelist.begin(), edgelist.end());

DSU s(V);

int ans = 0;

cout << "Following are the edges in the "

"constructed MST"

<< endl;

for (auto edge : edgelist) {int w = edge[0];

int x = edge[1];

int y = edge[2];

if (s.find(x) != s.find(y)) {

s.unite(x, y);

ans += w;

cout << x << " -- " << y << " == " << w

<< endl;

}

}

cout << "Minimum Cost Spanning Tree: " << ans;

}

};

int main()

{Graph g(4);

g.addEdge(0, 1, 10);

g.addEdge(1, 3, 15);

g.addEdge(2, 3, 4);

g.addEdge(2, 0, 6);

g.addEdge(0, 3, 5);g.kruskals\_mst();

return 0;

}