

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)

```
1. #include <bits/stdc++.h>
2. using namespace std;
3. const int mx = 100000;
4. pair <long long, pair<int, int> > store[mx];
5. int indx[mx], vertex, path;
6.
7. void init()
8. {
9.     for(int i = 0; i < mx; ++i)
10.         indx[i] = i;
11. }
12.
13. int pick_root(int t)
14. {
15.     while(indx[t] != t)
16.     {
17.         indx[t] = indx[indx[t]];
18.         t = indx[t];
19.     }
20.     return t;
21. }
22.
23. void union_func(int a, int b)
24. {
25.     int store = pick_root(a);
26.     int q = pick_root(b);
27.     indx[store] = indx[q];
28. }
29.
30. long long kruskal(pair<long long, pair<int, int> > store[])
31. {
32.     int a, b;
33.     long long cost, min_cost = 0;
34.     for(int i = 0; i < path; ++i)
35.     {
```

```

36.     a = store[i].second.first;
37.     b = store[i].second.second;
38.     cost = store[i].first;
39.     if(pick_root(a) != pick_root(b))
40.     {
41.         min_cost += cost;
42.         union_func(a, b);
43.     }
44. }
45. return min_cost;
46. }
47. int main()
48. {
49.     int v, e;
50.     long long w, cost, min_cost;
51.     init();
52.     cin >> vertex >> path;
53.     for(int i = 0; i < path; ++i)
54.     {
55.         cin >> v >> e >> w;
56.         store[i] = make_pair(w, make_pair(v, e));
57.     }
58.     sort(store, store + path);
59.
60.     auto d1 = chrono::steady_clock::now();
61.     min_cost = kruskal(store);
62.     auto d2 = chrono::steady_clock::now();
63.
64.     double difference = double(chrono::duration_cast <chrono::nanoseconds> (d2
        -d1).count());
65.
66.     cout << "Total Cost: " << min_cost << endl;
67.     cout << "The time Required for kruskal(path compression): " << difference/1000
        000 << " millisecond";
68.     return 0;
69. }
70.

```

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)

```
1. #include <bits/stdc++.h>
2. using namespace std;
3. int find(int i,vector<int>&parent)
4. {
5.     while (parent[i] != i)
6.         i = parent[i];
7.     return i;
8. }
9.
10. int main()
11. {
12.     int V,E;
13.     cin>>V>>E;
14.     int i,j;
15.     long long c[100][100]={INT_MAX};
16.     for(i=0;i<=V+1;i++)
17.     {
18.         for(j=0;j<=V+1;j++)
19.         {
20.             c[i][j]=INT_MAX;
21.         }
22.     }
```

```

23.
24.   for(int i = 0; i < E; i++)
25.   {
26.       int u, v, w;
27.       cin >> u >> v >> w;
28.       c[u][v] = w;
29.       c[v][u] = w;
30.
31.   }
32.   auto d1 = chrono::steady_clock::now();
33.   vector<int> parent(V);
34.   for(int i = 0; i < V; i++)
35.       parent[i] = i;
36.   int min_cost = 0;
37.
38.   int edge_count = 0;
39.   while (edge_count < E)
40.   {
41.       int min = INT_MAX, a = -1, b = -1;
42.       for (int i = 0; i <= V; i++)
43.       {
44.           for (int j = 0; j <= V; j++)
45.           {
46.               if (find(i, parent) != find(j, parent) && c[i][j] < min)
47.               {
48.                   min = c[i][j];
49.                   a = i;
50.                   b = j;
51.               }
52.           }
53.       }
54.       int aa = find(a, parent);
55.       int bb = find(b, parent);
56.       parent[aa] = bb;
57.       edge_count++;
58.       min_cost += min;
59.   }
60.   auto d2 = chrono::steady_clock::now();
61.   double difference = double(chrono::duration_cast < chrono::nanoseconds > (d2
    -d1).count());

```

```

62. cout<<"Total cost: "<<min_cost<<endl;
63. cout<<"The time Required for kruskal(without
    path compression): "<<difference/1000000<<" millisecond";
64. return 0;
65. }
66.

```

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

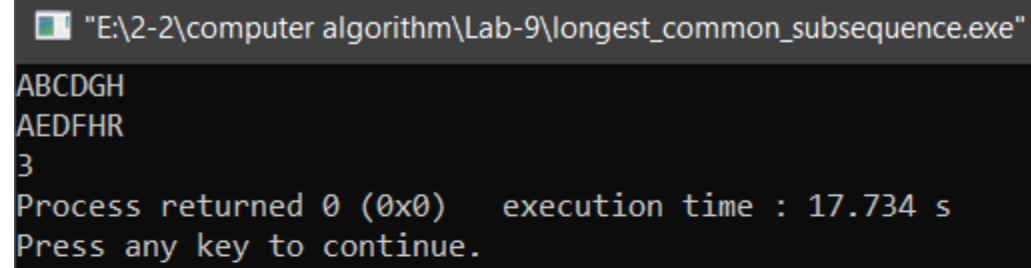
Problem Statement: Find the Longest Common Subsequence (LCS) between two strings using dynamic programming.

Code:

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. int Lcs(string x,string y)
4. {
5.     int m,n,i,j;
6.     m=x.length();
7.     n=y.length();
8.     int f_arr[m+1][n+1];
9.     for(i=0; i<=m; i++)
10.    {
11.        f_arr[0][i]=0;
12.    }
13.     for(i=0; i<=n; i++)
14.    {
15.        f_arr[i][0]=0;
16.    }
17.     for(i=1; i<=m; i++)
18.    {
19.        for(j=1; j<=n; j++)
20.        {
21.            if(x[i-1]==y[j-1])
22.            {
23.                f_arr[i][j]=f_arr[i-1][j-1]+1;
24.            }
25.            else
26.                f_arr[i][j]=max(f_arr[i][j-1],f_arr[i-1][j]);
27.        }
28.    }
29.     return f_arr[m][n];
30. }
31. int main()
32. {
33.     string a,b;
34.     cin>>a>>b;
```

```
35. cout<<Lcs(a,b);  
36. }
```

Output:



The screenshot shows a Windows command prompt window with a dark background. The title bar at the top reads "E:\2-2\computer algorithm\Lab-9\longest_common_subsequence.exe". The command prompt displays the following output: "ABCDGH", "AEDFHR", and the number "3". Below this, it shows "Process returned 0 (0x0) execution time : 17.734 s" and "Press any key to continue.".

```
"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.
```