**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

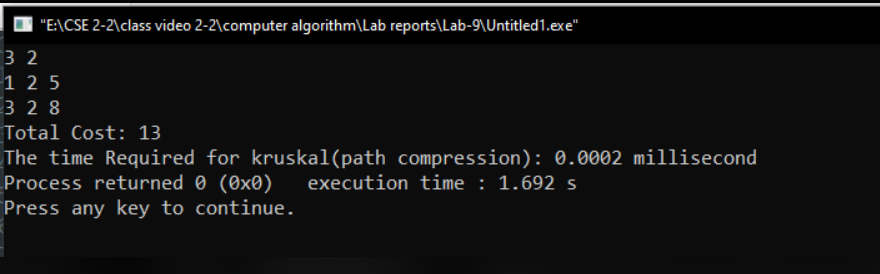
SUBMISSION DATE – 15 AUGUST, 2021

# 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;
7. void init()
8. {
9. for(int i = 0;i < mx;++i)
10. indx[i] = i;
11. }
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. }
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. }
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);
60. auto d1 = chrono::steady\_clock::now();
61. min\_cost = kruskal(store);
62. auto d2 = chrono::steady\_clock::now();
64. double difference = double(chrono::duration\_cast <chrono::nanoseconds> (d2-d1).count());
66. cout <<"Total Cost: "<< min\_cost << endl;
67. cout<<"The time Required for kruskal(path compression): "<<difference/1000000<<" millisecond";
68. return 0;
69. }

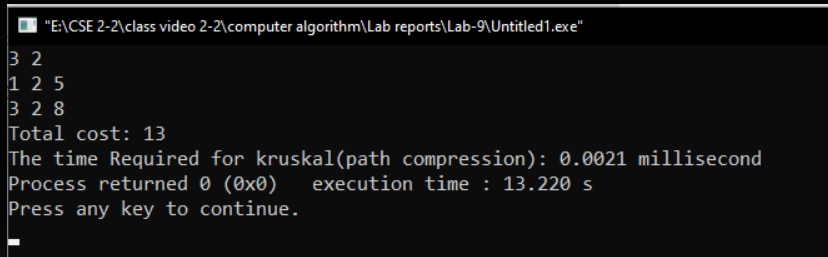
**Output:**



**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. }
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. }
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;
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;
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[a] = b;
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. }

**Output:**



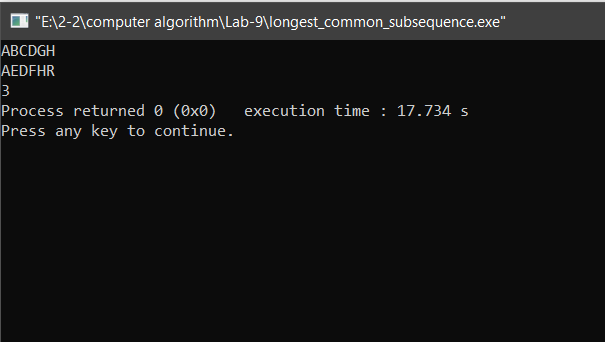
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **V** | **E** | **Time (Path compression/Rank based approach)** | **Time(Without path compression/rank approach)** | **Solution**  **(Total cost)** |
| 3 | 2 | 0.0002 | 0.0021 | 13 |
| 5 | 7 | 0.0005 | 0.009 | 9 |
| 9 | 14 | 0.0007 | 0.0277 | 37 |

# 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:**

****