Ex:-

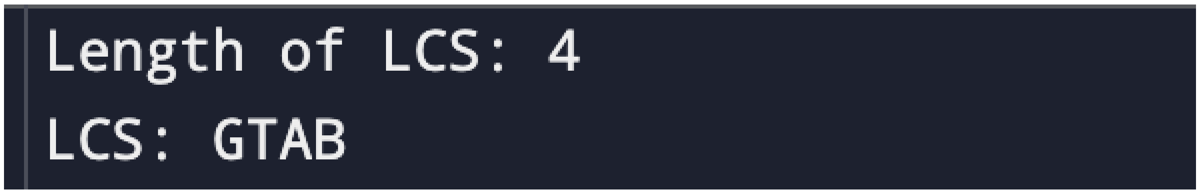
```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main() {
    string s1 = "AGGTAB", s2 = "GXTXAYB";
    int m = s1.size(), n = s2.size();
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (s1[i - 1] == s2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }

    int len = dp[m][n];
    string lcs(len, ' ');
    int i = m, j = n;
    while (i > 0 && j > 0) {
        if (s1[i - 1] == s2[j - 1]) {
            lcs[--len] = s1[i - 1];
            i--; j--;
        } else if (dp[i - 1][j] > dp[i][j - 1]) {
            i--;
        } else {
            j--;
        }
    }

    cout << "Length of LCS: " << dp[m][n] << endl;
    cout << "LCS: " << lcs << endl;
    return 0;
```



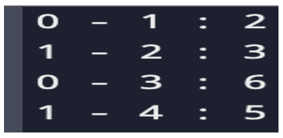
```
Length of LCS: 4
LCS: GTAB
```

Ex:-

```
#include <iostream>
#include <vector>
#include <climits>
using namespace std;

int findMinVertex(vector<int>& key, vector<bool>& mstSet, int V) {
    int minVal = INT_MAX, minIndex;
    for (int v = 0; v < V; v++) {
        if (!mstSet[v] && key[v] < minVal) {
            minVal = key[v];
            minIndex = v;
        }
    }
    return minIndex;
}
void primsAlgorithm(vector<vector<int>>& graph, int V) {
    vector<int> parent(V, -1), key(V, INT_MAX);
    vector<bool> mstSet(V, false);
    key[0] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = findMinVertex(key, mstSet, V);
        mstSet[u] = true;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }
 for (int i = 1; i < V; i++)
        cout << parent[i] << " - " << i << " : " << graph[i][parent[i]] << endl;
}
int main() {
    int V = 5;
    vector<vector<int>> graph = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},
        {0, 3, 0, 0, 7},
        {6, 8, 0, 0, 9},
        {0, 5, 7, 9, 0}
    };
    primsAlgorithm(graph, V);
    return 0;
}
```



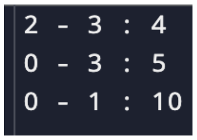
```
0 - 1 : 2
1 - 2 : 3
0 - 3 : 6
1 - 4 : 5
```

Ex:-

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Edge {
    int src, dest, weight;
};

bool compare(Edge a, Edge b) {
    return a.weight < b.weight;
}
int findParent(int v, vector<int>& parent) {
    if (parent[v] == v) return v;
    return parent[v] = findParent(parent[v], parent);
}
void unionSets(int u, int v, vector<int>& parent, vector<int>& rank) {
    u = findParent(u, parent);
    v = findParent(v, parent);
    if (rank[u] < rank[v])
        parent[u] = v;
    else if (rank[u] > rank[v])
        parent[v] = u;
    else {
        parent[v] = u;
        rank[u]++;
    }
}
void kruskalAlgorithm(vector<Edge>& edges, int V) {
    sort(edges.begin(), edges.end(), compare);
    vector<int> parent(V), rank(V, 0);
    for (int i = 0; i < V; i++) parent[i] = i;
    vector<Edge> mst;
    for (Edge e : edges) {
        int u = findParent(e.src, parent);
        int v = findParent(e.dest, parent);
        if (u != v) {
            mst.push_back(e);
            unionSets(u, v, parent, rank);
 }
}
 for (Edge e : mst)
        cout << e.src << " - " << e.dest << " : " << e.weight << endl;
}
int main() {
    int V = 4;
    vector<Edge> edges = {
        {0, 1, 10}, {0, 2, 6}, {0, 3, 5}, {1, 3, 15}, {2, 3, 4}
    };
    kruskalAlgorithm(edges, V);
    return 0;
}
```



```
2 - 3 : 4
0 - 3 : 5
0 - 1 : 10
```

Ex:

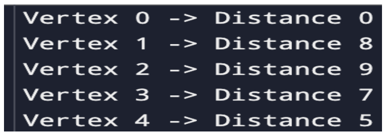
```
#include <iostream>
#include <vector>
#include <climits>
using namespace std;

int findMinVertex(vector<int>& dist, vector<bool>& visited, int V) {
    int minVal = INT_MAX, minIndex;
    for (int i = 0; i < V; i++) {
        if (!visited[i] && dist[i] < minVal) {
            minVal = dist[i];
            minIndex = i;
        }
    }
    return minIndex;
}

void dijkstraAlgorithm(vector<vector<int>>& graph, int V, int src) {
    vector<int> dist(V, INT_MAX);
    vector<bool> visited(V, false);
    dist[src] = 0;

    for (int i = 0; i < V - 1; i++) {
        int u = findMinVertex(dist, visited, V);
        visited[u] = true;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !visited[v] && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }

    for (int i = 0; i < V; i++)
        cout << "Vertex " << i << " -> Distance " << dist[i] << endl;
}
int main() {
    int V = 5;
    vector<vector<int>> graph = {
        {0, 10, 0, 0, 5},
        {0, 0, 1, 0, 2},
        {0, 0, 0, 4, 0},
        {7, 0, 6, 0, 0},
        {0, 3, 9, 2, 0}
    };
    dijkstraAlgorithm(graph, V, 0);
    return 0;
}
```



```
Vertex 0 -> Distance 0
Vertex 1 -> Distance 8
Vertex 2 -> Distance 9
Vertex 3 -> Distance 7
Vertex 4 -> Distance 5
```