```cpp
/*
    Name: Krushkal Algorithm
    Author: Sounish Nath
    Date: 17-09-20 18:23
    Description: To find minimal spanning tree from Graph
*/

#include <bits/stdc++.h>
using namespace std ;
typedef long long ll  ;

unordered_map<char, char> parent ; // check whos' parent
struct Edge{
    int cost ;
    char u, v ;
    Edge(char uu, char vv, int c) : cost(c), u(uu), v(vv) { }
};

struct Graph {
    vector<char> vertices ;
    vector<Edge> edges ;
};

struct Dset {
    void makeset(char u) {
        parent[u] = u ;
    }
    char find(char u) {
        if(parent[u] == u){
            return u ;
        }
        return find(parent[u]) ;
    }
    void unionn(char u, char v){
        char x = find(u) ;
        char y = find(v) ;
        parent[x] = y ;
    }
};

void krushhkal_algorithm(Graph &g) {
    Dset dset ;
    sort(g.edges.begin(), g.edges.end(), [&](Edge a, Edge b) {
        return a.cost < b.cost ;
    }) ;
    for(auto &&vertex : g.vertices){
        dset.makeset(vertex) ;
    }
    vector<Edge> a ;
    for(auto &&edge : g.edges) {
        char x = dset.find(edge.u)    ;
        char y = dset.find(edge.v)    ;
        if(x != y){
            a.push_back(edge) ;
            dset.unionn(x, y) ;
        }
    }
    int mincost = 0 ;
    cout << "going paths: \n" ;
    for(auto &&edge : a) {
        mincost += edge.cost ;
        cout << edge.u << " ---> " << edge.v << endl ;
    }
    cout << "minimum cost: " << mincost << endl ;
}

int main() {
    srand(time(NULL)) ;
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    char t[]{'a', 'b', 'c', 'd', 'e', 'f'};

    Graph g ;
    g.vertices = vector<char>(t, t+ sizeof(t)/sizeof(t[0])) ;
    g.edges.push_back(Edge('a', 'b', 2)) ;
    g.edges.push_back(Edge('b', 'd', 2)) ;
    g.edges.push_back(Edge('c', 'e', 7)) ;
    g.edges.push_back(Edge('f', 'a', 10)) ;
    g.edges.push_back(Edge('e', 'e', 1)) ;
    g.edges.push_back(Edge('d', 'f', 6)) ;
    g.edges.push_back(Edge('d', 'c', 9)) ;

    krushhkal_algorithm(g) ;


}

```