Dijkstra

```cpp
#include<bits/stdc++.h>
using namespace std;

# define INF 0x3f3f3f3f
typedef pair<int, int> intPair;

void addEdge(vector<intPair> adj[], int u, int v, int w){
    adj[u].push_back(make_pair(v, w));
    adj[v].push_back(make_pair(u, w));
}

void shortestPath(vector<intPair> adj[], int V, int src){
    priority_queue<intPair> pq;
    vector<int> dist(V, INF);

    pq.push(make_pair(0, src));
    dist[src] = 0;

    while (!pq.empty()){
        int u = pq.top().second;
        pq.pop();

        vector< intPair >::iterator i;
        for (i = adj[u].begin(); i != adj[u].end(); ++i){
            int v = (*i).first;
            int weight = (*i).second;

            if (dist[v] > dist[u] + weight){
                dist[v] = dist[u] + weight;
                pq.push(make_pair(dist[v], v));
            }
        }
    }

    printf("Vertex Distance from Source\n");
    for (int i = 0; i < V; ++i)
        printf("%d \t\t %d\n", i, dist[i]);
}

int main(){
    int V = 9;
    vector<intPair > adj[V];
    addEdge(adj, 0, 1, 4);
    addEdge(adj, 0, 7, 8);
    addEdge(adj, 1, 2, 8);
    addEdge(adj, 1, 7, 11);
    addEdge(adj, 2, 3, 7);
    addEdge(adj, 2, 8, 2);
    addEdge(adj, 2, 5, 4);
    addEdge(adj, 3, 4, 9);
    addEdge(adj, 3, 5, 14);
    addEdge(adj, 4, 5, 10);
    addEdge(adj, 5, 6, 2);
    addEdge(adj, 6, 7, 1);
    addEdge(adj, 6, 8, 6);
    addEdge(adj, 7, 8, 7);

    shortestPath(adj, V, 0);

    return 0;
}
```