Prims:

#include <bits/stdc++.h>  
using namespace std;  
  
// void implementPrims(vector<vector<pair<int,int>>>& adj,vector<int>& visited,priority\_queue<pair<int,int>>& pq,vector<int>& parent){  
//     if(pq.empty())  
//       return;  
//     int node=pq.front().second;  
//     visited[node]=1;  
//     for(auto a:adj[node]){  
//         if(!visited[a.second]){  
             
//         }  
//     }  
     
// }  
  
vector<pair<int,int>> prims(vector<vector<pair<int,int>>>& adj,int n,int start){  
    vector<int> visited(n+1);  
    vector<int> parent(n+1,-1);  
    priority\_queue<pair<pair<int,int>,int>,vector<pair<pair<int,int>,int>>,greater<pair<pair<int,int>,int>>> pq;  
    pq.push({{0,0},-1});  
    int ans=0;  
    vector<pair<int,int>> edges;  
     
    while(!pq.empty()){  
        int weight=pq.top().first.first;  
        int curr=pq.top().first.second;  
        int parent=pq.top().second;  
        pq.pop();  
        if(!visited[curr]){  
            visited[curr]=1;  
            ans+=weight;  
            if(parent != -1)  
               edges.push\_back({parent,curr});  
            for(auto node:adj[curr]){  
                int neighbor=node.second;  
                int dis=node.first;  
                if(!visited[neighbor])  
                   pq.push({{dis,neighbor},parent});  
                 
            }  
        }  
         
    }  
    cout<<ans;  
    return edges;  
}  
  
  
int main()  
{  
   int n,e,w,v1,v2,start;  
   cin>>n>>e;  
   vector<vector<pair<int,int>>> v(n+1,vector<pair<int,int>>());  
     
   for(int i=0;i<e;i++){  
       cin>>v1>>v2>>w;  
       v[v1].push\_back({w,v2});  
       v[v2].push\_back({w,v1});  
   }  
     
   for(int i=0;i<n;i++){  
       for(int j=0;j<v[i].size();j++){  
           cout<<i<<"->"<<v[i][j].second<<" = "<<v[i][j].first<<endl;  
       }  
   }  
   cin>>start;  
   vector<pair<int,int>> edges=prims(v,n,start);  
   cout<<endl;  
   for(auto edge:edges){  
       cout<<edge.first<<"->"<<edge.second<<"\n";  
   }  
    return 0;  
}

|  |  |
| --- | --- |
|  |  |