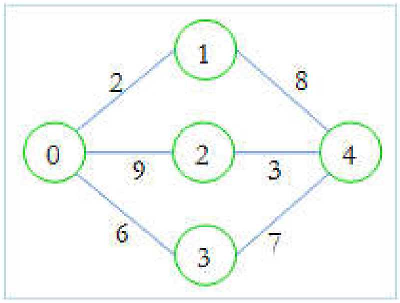
1002 - Country Roads

SUBMIT	PDF (English)	Statistics	Forum
Time Limit: 3 second(s)		Memory Limit: 32 MB	

I am going to my home. There are many cities and many bi-directional roads between them. The cities are numbered from **0** to **n-1** and each road has a cost. There are **m** roads. You are given the number of my city **t** where I belong. Now from each city you have to find the minimum cost to go to my city. The cost is defined by the cost of the maximum road you have used to go to my city.



For example, in the above picture, if we want to go from 0 to 4, then we can choose

1) 0 - 1 - 4 which costs 8, as 8 (1 - 4) is the maximum road we used
2) 0 - 2 - 4 which costs 9, as 9 (0 - 2) is the maximum road we used
3) 0 - 3 - 4 which costs 7, as 7 (3 - 4) is the maximum road we used

So, our result is 7, as we can use 0 - 3 - 4.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a blank line and two integers **n (1 ≤ n ≤ 500)** and **m (0 ≤ m ≤ 16000)**. The next **m** lines, each will contain three integers **u, v, w (0 ≤ u, v < n, u ≠ v, 1 ≤ w ≤ 20000)** indicating that there is a road between **u** and **v** with cost **w**. Then there will be a single integer **t (0 ≤ t < n)**. There can be multiple roads between two cities.

Output

For each case, print the case number first. Then for all the cities (from **0** to **n-1**) you have to print the cost. If there is no such path, print **'Impossible'**.

Sample Input	Output for Sample Input

2
 	
5 6
0 1 5
0 1 4
2 1 3
3 0 7
3 4 6
3 1 8
1
 	
5 4
0 1 5
0 1 4
2 1 3
3 4 7
1
	Case 1:
	4
	0
	3
	7
	7
	Case 2:
	4
	0
	3
	Impossible
	Impossible

```cpp
/// 1002 - Country Roads
#define inf 1000000000
int n,m,d[505],cost[505][505]; vector<int>ed[505];
struct edge{
    int u,v,w;
    edge(int x, int y, int c){ u=x;v=y;w=c; }
    bool operator < (const edge& p) const{ return p.w < w; }
};
void primsMST(int s){
    priority_queue<edge>pq;
    for(int i=0; i<ed[s].size(); i++){
        pq.push(edge(s,ed[s][i],cost[s][ed[s][i]]));
    }
    memset(d,-1,sizeof(d)); d[s]=0;

    while(!pq.empty()){
        edge nd(-1,-1,-1); nd = pq.top(); pq.pop();
        int u=nd.u, v=nd.v, w=nd.w;
        if(d[v]!=-1) continue; // creating cycle
        d[v] = max(d[u],w);
        for(int i=0; i<ed[v].size(); i++){
            if(d[ed[v][i]]==-1){
                pq.push(edge(v,ed[v][i],cost[v][ed[v][i]]));
            }
        }
    }
}
int main(){
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++){
        scanf("%d%d",&n,&m);
        for(int i=0; i<=n; i++){
            for(int j=0; j<=n; j++){
                cost[i][j]=inf;
            }
        }

        for(int i=1; i<=m; i++){
            int u,v,w; scanf("%d%d%d",&u,&v,&w);
            if(cost[u][v]==inf){
                ed[u].push_back(v); ed[v].push_back(u);
            }
            cost[u][v]=cost[v][u]=min(cost[u][v],w);
        }
        int s; scanf("%d",&s);

        primsMST(s);

        printf("Case %d:\n",ks);
        for(int i=0; i<n; i++){
            if(d[i]!=-1)printf("%d\n",d[i]);
            else printf("Impossible\n");
        }
        for(int i=0; i<=n; i++)ed[i].clear();
    }
}
/** Another Solution
This problem wants you to find the minimum cost to reach every city from a given city(my city).
Here minimum cost signifies the maximum road cost along the path from a city to another city,
you have to minimize this cost. This problem can easily be solved using modified dijkstra.
Set the distance of the start node(my city t) to 0 and then run dijkstra from it. Instead of
adding the road cost to the node in the relax function, you have to relax the current node
using the maximum road cost since you have to minimize the maximum road cost in the path.
The relax function can be written like this:
//here u is the source node and v is the destination node and w is the road cost from u to v
and dis array stores the cost of the node upon which we'll relax
        if( dis[ v ] > max( w , dis[ u ] ) ) {
                dis[ v ] = max( w , dis[ u ] ) ;
        }
Time Complexity: O(E * lg V). E is the number of roads and V is the number of cities.
**/
```