**Dijkstra with finding Kth shortest path , weighted, O(E+V logV) for k=1**

**Shortest[k][v] contain the kth shortest path to v**

**Tested on: LightOJ 1099 – Not the best**

struct Z{

int v, cost;

bool operator <(const Z &p)const{

return cost>p.cost;

}

};

Z z, z2;

int shortest[5][MX]; //shortest[k+][node+]

vector<Z>adj[MX];

priority\_queue<Z>pq;

int main()

{

int kk=1, tc, n, m, node, edge, u, v, cost, vcost, src, dst, k;

string s;

cin>>tc;

while(tc--)

{

k=2;

cin>>node>>edge;

for(int i=1;i<=node;i++)

{

for(int j=1;j<=k;j++) shortest[j][i]=1e9;

shortest[0][i]=-1e9;

adj[i].clear();

}

for(int i=0;i<edge;i++)

{

cin>>u>>v>>cost;

z.v=u, z.cost=cost;

adj[v].push\_back(z);

z.v=v, z.cost=cost;

adj[u].push\_back(z);

}

src=1, dst=node;

z.v=src, z.cost=0, shortest[1][src]=0;

pq.push(z);

/\*for(int i=1;i<=k;i++) //comment it if edge can revisit.

shortest[i][src]=0;\*/

while(!pq.empty())

{

z=pq.top();

pq.pop();

int l=adj[z.v].size();

for(int i=0;i<l;i++)

{

z2=adj[z.v][i];

v=z2.v, vcost=z2.cost;

for(int j=1;j<=k;j++)

if(shortest[j-1][v]<z.cost+vcost && shortest[j][v]>z.cost+vcost)

{

shortest[j+1][v]=shortest[j][v];

z2.cost=shortest[j][v]=z.cost+vcost;

pq.push(z2);

break;

}

}

}

cout<<"Case "<<kk++<<": "<<shortest[k][dst]<<"\n";

}

return 0;

}