**//Directed MST**

/\*

Dfs to check whether solution exists or not. Traverse(source) for directed mst.

nv is total node, ne total edge.

Complexity O(V\*E)

\*/

int nv, ne, res, vis[MX], pre[MX], num[MX], inLength[MX]; //MX =maximum node

vector<int>adj[MX];

struct Edge{

int u, v, w;

}edge[10\*MX]; //maximum edge

int Traverse(int root)

{

while(true)

{

for(int i=0; i<nv; i++) inLength[i] = 1e9;

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

{

int u = edge[i].u;

int v = edge[i].v;

if(edge[i].w < inLength[v] && u != v)

{

inLength[v] = edge[i].w;

pre[v] = u;

}

}

int newnum = 0;

inLength[root] = 0;

SET(vis);

SET(num);

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

{

res += inLength[i];

int v = i;

while(vis[v] != i && num[v] == -1 && v != root)

{

vis[v] = i;

v = pre[v];

}

if(vis[v] == i)

{

for(int u=pre[v]; u != v; u = pre[u])

num[u] = newnum;

num[v] = newnum++;

}

}

if(newnum == 0) return res;

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

if(num[i] == -1) num[i] = newnum++;

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

{

int u = edge[i].u;

int v = edge[i].v;

edge[i].u = num[u];

edge[i].v = num[v];

if(edge[i].u != edge[i].v)

edge[i].w -= inLength[v];

}

nv = newnum;

root = num[root];

}

return res;

}