最小瓶颈： 树上路径倍增 + LCA

#include<cstdio>

#include<algorithm>

#define ll long long

#define MaxM 50010

#define MaxN 10010

#define MaxQ 100000

int n,m,q,x,y,tot=0;

int fa[MaxN],Clock[MaxN];

int v[MaxN\*2],w[MaxN\*2],head[MaxN],next[MaxN\*2];

int f[MaxN][14],g[MaxN][14];

//f[i,j]记录i结点向上走2^j步后所到达的祖先

//g[i,j]记录i结点向上走2^j步的路途中的边权最小值

struct TypeEdge{int u,v,w;}ed[MaxM];

bool cmp(TypeEdge a,TypeEdge b){return a.w>b.w;}//边权从大到小

int find(int x){return fa[x]==x?fa[x]:fa[x]=find(fa[x]);}

void dfs(int now,int clock\_now)

{

Clock[now]=clock\_now; //在dfs树上的深度

for (int i=1;i<=13;i++)

{

f[now][i]=f[f[now][i-1]][i-1];

g[now][i]=min(g[now][i-1],g[f[now][i-1]][i-1]);

}

for (int i=head[now];i;i=next[i])

if (!Clock[v[i]])

{

f[v[i]][0]=now;

g[v[i]][0]=w[i];

dfs(v[i],clock\_now+1);

}

}

int LCA(int a,int b)

{

if (Clock[a]>Clock[b]) swap(a,b);

int Ans=MaxQ+1;

for (int i=13;i+1;i--)

if (Clock[f[b][i]]>=Clock[a])

{

Ans=min(Ans,g[b][i]);

b=f[b][i];

}//将ab移动至同一深度

if (a==b) return Ans;

for (int i=13;i+1;i--)

if (f[a][i]!=f[b][i])

{

Ans=min(Ans,min(g[a][i],g[b][i]));

a=f[a][i];b=f[b][i];

}//向上找到LCA

Ans=min(Ans,min(g[a][0],g[b][0]));

return Ans;

}

int main()

{

scanf("%d%d",&n,&m);

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

scanf("%d%d%d",&ed[i].u,&ed[i].v,&ed[i].w);

sort(ed+1,ed+m+1,cmp);

for (int i=1;i<=n;i++) fa[i]=i;

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

{

int x=find(ed[i].u),y=find(ed[i].v);//最优生成树

if (x!=y)

{

v[++tot]=ed[i].v;w[tot]=ed[i].w;next[tot]=head[ed[i].u];head[ed[i].u]=tot;

v[++tot]=ed[i].u;w[tot]=ed[i].w;next[tot]=head[ed[i].v];head[ed[i].v]=tot;

fa[x]=y;

}

}

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

if (!Clock[i]) dfs(i,1);

scanf("%d",&q);

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

{

scanf("%d%d",&x,&y);

if (find(x)!=find(y)) puts("-1");

else printf("%d\n",LCA(x,y));

}

return 0;

}