上下界网络流

//可二分t->s边的下/上界,即可达到最大最小流

//最大流:t->s连边,ss->tt流,s->t正向最大流,会流掉反向建的边的流量

//最小流:ss->tt流,t->s连边,ss->tt流

int n,m,q;

int i,j,k;

int ss,tt;

struct node{

int to,cap,next;

}edge[maxn\*3];

int tot;

int head[307];

int addedge(int from,int to,int cap){

edge[tot].to=to;

edge[tot].next=head[from];

edge[tot].cap=cap;

head[from]=tot++;

edge[tot].to=from;

edge[tot].next=head[to];

edge[tot].cap=0;

head[to]=tot++;

return tot-1;//反的边 cap=正的 flow

}

bool vis[307];

int d[307];

queue<int> Q;

bool bfs(int s,int t){

memset(vis,0,sizeof(vis));

while (Q.size()) Q.pop();

Q.push(s);

d[s]=0;vis[s]=1;

int i;

while (Q.size()){

int x=Q.front();Q.pop();

for (i=head[x];i!=-1;i=edge[i].next){

if (!vis[edge[i].to]&&edge[i].cap){

vis[edge[i].to]=1;

d[edge[i].to]=d[x]+1;

Q.push(edge[i].to);

}

}

}

return vis[t];

}

int cur[307];//当前弧优化

int dfs(int x,int t,int flow){//dinic

if (x==t||flow==0) return flow;

int i,ret=0,f;

for (i=cur[x];i!=-1;i=edge[i].next){

if (d[x]+1==d[edge[i].to]&&(f=dfs(edge[i].to,t,min(flow,edge[i].cap)))>0){

edge[i].cap-=f;

edge[i^1].cap+=f;

ret+=f;

flow-=f;

cur[x]=i;

if (flow==0) break;

}

}

return ret;

}

int in[307],out[307];

int add(int u,int v,int low,int high){

int ret=addedge(u,v,high-low);

out[u]+=low;in[v]+=low;

return ret;

}

int sum,flow,E[maxn],ans[maxn];//E为对应的边位置

int solve(){

memset(head,0xff,sizeof(head));

memset(in,0,sizeof(in));

memset(out,0,sizeof(out));

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

flow=0;sum=0;tot=0;

FOR(i,1,m){

int u,v,low,high;

scanf("%d%d%d%d",&u,&v,&low,&high);

ans[i]=low;

E[i]=add(u,v,low,high);//E[i]很有用

}

ss=n+1;tt=n+2;

FOR(i,1,n){

sum+=max(in[i]-out[i],0);

if (in[i]>out[i]) addedge(ss,i,in[i]-out[i]);

if (in[i]<out[i]) addedge(i,tt,out[i]-in[i]);

}

while (bfs(ss,tt)){

int f;

memcpy(cur,head,sizeof(head));

while (f=dfs(ss,tt,INF)) flow+=f;

}

if (flow!=sum) return 0\*puts("NO");

else {

puts("YES");

FOR(i,1,m){

ans[i]+=edge[E[i]].cap;

printf("%d\n",ans[i]);

}

}

}

int main()

{

int T;

scanf("%d",&T);

while (T--){

solve();

}

}