// 最大流 dinic

using namespace std;

const int Ni = 210;

const int MAX = 1<<26;

struct Edge{

int u,v,c;

int next;

}edge[20\*Ni];

int n,m;

int edn;//边数

int p[Ni];//父亲

int d[Ni];

int sp,tp;//原点，汇点

void addedge(int u,int v,int c)

{

edge[edn].u=u; edge[edn].v=v; edge[edn].c=c;

edge[edn].next=p[u]; p[u]=edn++;

edge[edn].u=v; edge[edn].v=u; edge[edn].c=0;

edge[edn].next=p[v]; p[v]=edn++;

}

int bfs()

{

queue <int> q;

memset(d,-1,sizeof(d));

d[sp]=0;

q.push(sp);

while(!q.empty())

{

int cur=q.front();

q.pop();

for(int i=p[cur];i!=-1;i=edge[i].next)

{

int u=edge[i].v;

if(d[u]==-1 && edge[i].c>0)

{

d[u]=d[cur]+1;

q.push(u);

}

}

}

return d[tp] != -1;

}

int dfs(int a,int b)

{

int r=0;

if(a==tp)return b;

for(int i=p[a];i!=-1 && r<b;i=edge[i].next)

{

int u=edge[i].v;

if(edge[i].c>0 && d[u]==d[a]+1)

{

int x=min(edge[i].c,b-r);

x=dfs(u,x);

r+=x;

edge[i].c-=x;

edge[i^1].c+=x;

}

}

if(!r)d[a]=-2;

return r;

}

int dinic(int sp,int tp)

{

int total=0,t;

while(bfs())

{

while(t=dfs(sp,MAX))

total+=t;

}

return total;

}

int main()

{

int i,u,v,c;

while(~scanf("%d%d",&m,&n))

{

edn=0;//初始化

memset(p,-1,sizeof(p));

sp=1;tp=n;

for(i=0;i<m;i++)

{

scanf("%d%d%d",&u,&v,&c);

addedge(u,v,c);

}

printf("%d\n",dinic(sp,tp));

}

return 0;

}

// 最大流2

const int maxn=1000+5, bign=1000000000;

int M, n, m, source, sink, c[maxn][maxn], cnt[maxn];

/\* The arc of the flow network.\*/

struct Pool

{

int next, t, c;

} edge[maxn\*maxn<<1];

/\* The point of the flow network.\*/

struct Point

{

int son, cur, pre, lim, d;

} a[maxn];

/\* Prepare for the algorithm.\*/

void initialize()

{

M=1;

memset(c, 0, sizeof (c));

memset(a, 0, sizeof (a));

memset(cnt, 0, sizeof (cnt));

}

/\* Add an arc to the flow network.\*/

void add(int x, int y, int z)

{

edge[++M].t=y;

edge[M].c=z;

edge[M].next=a[x].son;//相当于pool的head数组

a[x].son=M;

}

/\* Read the data and make it the right format.\*/

void input()

{

scanf("%\*s%\*d%d%d%d%d", &n, &m, &source, &sink);

initialize();

int x, y, z;

while (m--)

scanf("%d%d%d", &x, &y, &z), c[x][y]+=z;

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

for (int j=0; j<n; j++)

if (c[i][j])add(i, j, c[i][j]),

add(j, i, c[j][i]),

c[j][i]=0;

}

int que[maxn], fi, la;

bool vis[maxn];

/\* Build the hierarchical graph for the algorithm\*/

bool build()

{

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

que[fi=la=0]=sink;//reverse

a[sink].d=0, a[sink].cur=a[sink].son, vis[sink]=true;

while (fi<=la)

{

int v=que[fi++];

for (int now=a[v].son, u; u=edge[now].t, now; now=edge[now].next)

if (edge[now^1].c&&!vis[u])//BFS来分层,这里和EK相同

{//倒着BFS的话,当然引用的还是对侧边,即正向边

a[u].d=a[v].d+1;//越向前标号渐大

a[u].cur=a[u].son;//cur指向头

vis[u]=true;//已遍历

que[++la]=u;//入队

}

if (vis[source])return true;//层次图向前已经扩展到原点

}

return false;

}

/\*Use the Dinic algorithm to calculate the max flow.\*/

int MaxFlow()

{

int u, v, now, ret=0;

while (build())

{

a[u=source].lim=bign;

while (true)

{

for (now=a[u].cur; v=edge[now].t, now; now=edge[now].next)//cur优化

if (edge[now].c&&a[u].d==a[v].d+1)break;//找到了一个子节点属于层次图

if (now)

{

a[u].cur=edge[now].next;//下一次从这一条边的下一条边开始dfs

a[v].pre=now;//指向v的边的指针

a[v].lim=min(a[u].lim, edge[now].c);///更新到此处为止流的上限

if ((u=v)==sink)//如果已经找到了一条增广路(走到了尽头)

///注意这个地方借判断语句, 将u下移, 便于判断为否的时候回到上面进入下一层!

{//进行增广

do

{

edge[a[u].pre].c-=a[sink].lim;

edge[a[u].pre^1].c+=a[sink].lim;//这两句和Edmonds-Karp是一样的,增广

u=edge[a[u].pre^1].t;//找前驱~!

} while (u!=source);

ret+=a[sink].lim;//增广完毕之后累加新找到的流

}//否则(没走到尽头)继续向下DFS

}

else//没有子节点属于层次图

{

if (u==source)break;//已经退到了源,则已找到最大流,算法结束

a[u].cur=now;//=0,此节点被废弃,子代亦然

u=edge[a[u].pre^1].t;//根据反向边找到前驱~!

}

}

}

return ret;

}

int main()

{

int total;

scanf("%d", &total);

while (total--)

{

input();

printf("%d\n", MaxFlow());

}}