// 网络流全家桶

namespace Flow

{

//dinic求最大流

int t,tot=1,head[1048],cur[1048],to[400048],nxt[400048],f[400048],cost[400048];

inline void addedge(int s,int t,int cap)

{

to[++tot]=t;nxt[tot]=head[s];head[s]=tot;f[tot]=cap;

to[++tot]=s;nxt[tot]=head[t];head[t]=tot;f[tot]=0;

}

int depth[1048];queue<int> q;

bool bfs()

{

int i,x,y;

for (i=0;i<=t;i++) depth[i]=-1;

depth[0]=0;q.push(0);

while (!q.empty())

{

x=q.front();q.pop();

for (i=head[x];i;i=nxt[i])

{

y=to[i];

if (f[i] && depth[y]==-1)

{

depth[y]=depth[x]+1;

q.push(y);

}

}

}

if (depth[t]==-1) return false; else return true;

}

int dfs(int x,int maxf)

{

if (x==t) return maxf;

int y,now,ans=0,minf;

for (int &i=cur[x];i;i=nxt[i])

{

y=to[i];

if (f[i] && depth[y]==depth[x]+1)

{

minf=min(f[i],maxf-ans);

now=dfs(y,minf);

f[i]-=now;

f[i^1]+=now;

ans+=now;

if (ans>=maxf) return ans;

}

}

if (ans==0) depth[x]=0;

return ans;

}

inline int dinic()

{

int ans=0;

while (bfs())

{

for (register int i=0;i<=t;i++) cur[i]=head[i];

ans+=dfs(0,INF);

}

return ans;

}

//zkw费用流

inline void addedge(int s,int t,int cap,int c)

{

to[++tot]=t;nxt[tot]=head[s];head[s]=tot;f[tot]=cap;cost[tot]=c;

to[++tot]=s;nxt[tot]=head[t];head[t]=tot;f[tot]=0;cost[tot]=-c;

}

int n,e,S,T;

int D[200048];bool visited[10048];

int maxflow,mincost;

inline int aug(int cur,int maxf)

{

visited[cur]=true;

if (cur==T) {maxflow+=maxf;mincost+=(-D[S])\*maxf;return maxf;}

int y,now,minf,ans=0;

for (int &i=head[cur];i;i=nxt[i])

{

y=to[i];

if (f[i] && !visited[y] && D[cur]+cost[i]-D[y]==0)

{

minf=min(maxf-ans,f[i]);

now=aug(y,minf);

f[i]-=now;f[i^1]+=now;ans+=now;

if (ans>=maxf) return ans;

}

if (ans>=maxf) return ans;

}

return ans;

}

inline bool relabel()

{

int d=INF,cur,i,y;

for (cur=1;cur<=n;cur++)

if (visited[cur])

for (i=head[cur];i;i=nxt[i])

{

y=to[i];

if (f[i] && !visited[y]) d=min(d,D[cur]+cost[i]-D[y]);

}

if (d>=INF) return false;

for (cur=1;cur<=n;cur++) if (visited[cur]) D[cur]-=d;

return true;

}

inline int zkw()

{

do

{

for (i=1;i<=n;i++) cur[i]=head[i];

do

memset(visited,false,sizeof(visited));

while (aug(S,INF));

}

while (relabel());

return maxflow;

}

//edmonds-karp-dijkstra 费用流

int prevv[200048],preve[200048];

int h[200048];

priority\_queue<Pair> q;int dist[200048];

bool dijkstra()

{

int i,x,y,dd;

for (i=1;i<=n;i++) dist[i]=INF;

dist[s]=0;q.push(mp(0,s));

while (!q.empty())

{

x=q.top().y;dd=-q.top().x;q.pop();

if (dd>dist[x]) continue;

for (i=head[x];i;i=nxt[i])

{

y=to[i];

if (f[i] && dist[x]+cost[i]+h[x]-h[y]<dist[y])

{

dist[y]=dist[x]+cost[i]+h[x]-h[y];

prevv[y]=x;preve[y]=i;

q.push(mp(-dist[y],y));

}

}

}

if (dist[t]>=INF) return false; else return true;

}

queue<int> qq;bool inq[400048];

inline void spfa()

{

int i,x,y;

for (i=start;i<=end;i++) dist[i]=INF;

dist[start]=0;

for (i=start;i<=end;i++) inq[i]=false;

inq[start]=true;qq.push(start);

while (!qq.empty())

{

x=qq.front();qq.pop();inq[x]=false;

for (i=head[x];i;i=nxt[i])

{

y=to[i];

if (f[i] && dist[y]>dist[x]+w[i]+h[x]-h[y])

{

dist[y]=dist[x]+w[i]+h[x]-h[y];

prevv[y]=x;preve[y]=i;

if (!inq[y]) qq.push(y);

}

}

}

}

//第一维是最大流，第二维是最小费用

Pair min\_cost\_flow()

{

int i,u,res,minf;

for (i=1;i<=n;i++) h[i]+=dist[i];

minf=INF;

for (u=t;u!=s;u=prevv[u]) minf=min(minf,f[preve[u]]);

res=minf\*h[t];

for (u=t;u!=s;u=prevv[u])

{

f[preve[u]]-=minf;

f[preve[u]^1]+=minf;

}

return mp(minf,res);

}

inline int edmonds\_karp\_dijkstra()

{

Pair ans;

while (dijkstra())

{

Pair res=min\_cost\_flow();

ans.x+=res.x;ans.y+=res.y;

}

return ans.x;

}

inline int edmonds\_karp\_spfa\_dijkstra()

{

int ans=0;

spfa();

do

ans+=min\_cost\_flow();

while (dijkstra());

return ans;

}

}