森林中两棵树连起来的直径期望

vector<int> edge[maxn];

vector<int> value[maxn];

int len[maxn],root[maxn],num[maxn];

int mx,mxlen;

void dfs1(int u,int x,int length){//需要好多次(findmaxlen)

int i;

if (length>len[u]) len[u]=length;

if (length>mxlen) mx=u,mxlen=length;

REP(i,edge[u].size())

if (edge[u][i]!=x) dfs1(edge[u][i],u,length+1);

}

void dfs2(int x,int father){

int i;

root[x]=father;

value[father].push\_back(len[x]);

num[father]++;

REP(i,edge[x].size())

if (!root[edge[x][i]]) dfs2(edge[x][i],father);

}

map<pair<int,int>,double> H;

double solve(int x,int y){

x=root[x];y=root[y];

if (value[x].size()>value[y].size()) swap(x,y);

if (H.count({x,y})) return H[{x,y}];

LL sum=0,k,all=0;

int i,j,n,m;

k=max(len[x],len[y]);

n=value[x].size();

m=value[y].size();

j=m;

all=0;

REP(i,n){

while (j<m&&value[x][i]+value[y][j]+1<=k) {sum-=value[y][j];j++;}

while (j>0&&value[x][i]+value[y][j-1]+1>k) {j--;sum+=value[y][j];}

all+=sum+(value[x][i]+1)\*(m-j)+k\*j;

}

H[{x,y}]=1.0\*all/n/m;

H[{y,x}]=1.0\*all/n/m;

return 1.0\*all/n/m;

}

int n,m,q;

int i,j,k;

int u,v;

LL ans;

int main(){

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

REP(i,m){

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

edge[u].push\_back(v);

edge[v].push\_back(u);

}

FOR(i,1,n) if (!root[i]){

mxlen=-1;dfs1(i,0,0);u=mx;

mxlen=-1;dfs1(u,0,0);v=mx;

dfs1(v,0,0);

dfs2(u,u);

sort(value[u].begin(),value[u].end());

}

while (q--){

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

if (root[u]==root[v]) puts("-1");

else printf("%.10lf\n",solve(u,v));

}

}

/\*

\*/

#include <cstdio> 最小费用流DIJ

#include <vector>

#include <algorithm>

#include <cstring>

#include <queue>

using namespace std;

#define REP(I,N) for (I=0;I<N;I++)

#define rREP(I,N) for (I=N-1;I>=0;I--)

#define rep(I,S,N) for (I=S;I<N;I++)

#define rrep(I,S,N) for (I=N-1;I>=S;I--)

#define FOR(I,S,N) for (I=S;I<=N;I++)

#define rFOR(I,S,N) for (I=N;I>=S;I--)

typedef unsigned long long ULL;

typedef long long LL;

const int INF=0x3f3f3f3f;

const LL INFF=0x3f3f3f3f3f3f3f3fll;

const LL M=1e9+7;

const LL maxn=1e5+7;

const double eps=0.00000001;

LL gcd(LL a,LL b){return b?gcd(b,a%b):a;}

template<typename T>inline T abs(T a) {return a>0?a:-a;}

template<typename T>inline T powMM(T a,T b){T ret=1;for (;b;b>>=1ll,a\*=a) ret=1ll\*ret\*a%M;return ret;}

#define x x\_x

#define y y\_y

struct node{

LL to,cap,cost,rev;

node(int t=0,int c=0,int n=0,int r=0):to(t),cap(c),cost(n),rev(r){}

};

vector<node> edge[maxn];

void addedge(int from,int to,LL cap,LL cost){

edge[from].push\_back(node(to,cap,cost,edge[to].size()));

edge[to].push\_back(node(from,0,-cost,edge[from].size()-1));

}

int n,m,V;

LL dis[maxn],h[maxn];

int pre\_v[maxn],pre\_e[maxn];

priority\_queue<pair<LL,int> > Q;

pair<LL,LL> mincostflow(int s,int t,LL f){

LL ret=0,d;

int i,v;

memset(h,0,sizeof(h));//顶点的势

while (f){

memset(dis,0x3f,sizeof(dis));

Q.push(make\_pair(0ll,s));

dis[s]=0;

while (!Q.empty()){

pair<LL,int> y=Q.top();Q.pop();

v=y.second;

if (dis[v]<y.first) continue;

REP(i,edge[v].size()){

node &e=edge[v][i];

if (e.cap>0&&dis[e.to]>dis[v]+e.cost+h[v]-h[e.to]){

dis[e.to]=dis[v]+e.cost+h[v]-h[e.to];

pre\_v[e.to]=v;

pre\_e[e.to]=i;

Q.push(make\_pair(dis[e.to],e.to));

}

}

}

if (dis[t]==INFF) break;

REP(v,V) h[v]+=dis[v];

d=f;

for (v=t;v!=s;v=pre\_v[v])

d=min(d,edge[pre\_v[v]][pre\_e[v]].cap);

f-=d;

ret+=d\*dis[t];

for (v=t;v!=s;v=pre\_v[v]){

node &e=edge[pre\_v[v]][pre\_e[v]];

e.cap-=d;

edge[v][e.rev].cap+=d;

}

if (d==0) break;

}

return make\_pair(INFF-f,ret);

}

int i,j,k;

int main(){

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

FOR(i,1,m){

LL u,v,c,w;

scanf("%lld%lld%lld%lld",&u,&v,&c,&w);

addedge(u,v,c,w);

}V=n;

pair<LL,LL> ans=mincostflow(1,n,INFF);

printf("%lld %lld",ans.first,ans.second);

}

SPFA

#define x x\_x

#define y y\_y

struct node{

LL to,cap,cost,rev;

node(int t=0,int c=0,int n=0,int r=0):to(t),cap(c),cost(n),rev(r){}

};

vector<node> edge[maxn];

void addedge(int from,int to,LL cap,LL cost){

edge[from].push\_back(node(to,cap,cost,edge[to].size()));

edge[to].push\_back(node(from,0,-cost,edge[from].size()-1));

}

int n,m,V;

LL dis[maxn];

bool mark[maxn];

int pre\_v[maxn],pre\_e[maxn];

deque<int> Q;

pair<LL,LL> mincostflow(int s,int t,LL f){

LL ret=0,d;

int i,v;

while (f){

memset(dis,0x3f,sizeof(dis));

memset(mark,0,sizeof(mark));

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

dis[s]=0;Q.push\_back(s);

while (Q.size()){

v=Q.front();mark[v]=0;Q.pop\_front();

REP(i,edge[v].size()){

node &e=edge[v][i];

if (e.cap>0&&dis[e.to]>dis[v]+e.cost){

dis[e.to]=dis[v]+e.cost;

pre\_v[e.to]=v;

pre\_e[e.to]=i;

if (!mark[e.to]){

if (Q.empty()||dis[Q.front()]<dis[e.to]) Q.push\_back(e.to);

else Q.push\_front(e.to);

mark[e.to]=1;

}

}

}

}

if (dis[t]==INFF) break;

d=f;

for (v=t;v!=s;v=pre\_v[v])

d=min(d,edge[pre\_v[v]][pre\_e[v]].cap);

f-=d;

ret+=d\*dis[t];

for (v=t;v!=s;v=pre\_v[v]){

node &e=edge[pre\_v[v]][pre\_e[v]];

e.cap-=d;

edge[v][e.rev].cap+=d;

}

if (d==0) break;

}

return make\_pair(INFF-f,ret);

}

int i,j,k;

int main(){

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

FOR(i,1,m){

LL u,v,c,w;

scanf("%lld%lld%lld%lld",&u,&v,&c,&w);

addedge(u,v,c,w);

}V=n;

pair<LL,LL> ans=mincostflow(1,n,INFF);

printf("%lld %lld",ans.first,ans.second);

}