最小树形图

最小树形图：就是有向图中的最小生成树。

主要过程：

①：寻找最短入边集合in[i]；

②：检查入边集合：

1. 若集合中没有环，则算法结束。
2. 若含有环则缩点。

③：反复计算知道得出最小树形图为止；

缩点后注意的是，对于缩后的图要进重新建造，在原图中有指向该环中的Xi 的时候，那么这条边就要减去in[Xi];

\*注：对于一幅无根的图，我们需要加一个虚拟的根，这个虚拟的根到原图中的任意一点的距离大于原图中所有距离之和。

下面附上代码：

int Zhuliu(int root,int totn,int totm){

int x,y,ans=0;

while(true){

rep(i,0,totn) in[i]=inf;

rep(i,0,totm-1){

Edge& e=ed[i];

if(e.from!=e.to&&e.val<in[e.to]) {

in[e.to]=e.val;

pre[e.to]=e.from;

if(e.from==root) rt=i;

}

}

rep(i,0,totn) if(in[i]==inf&&i!=root) return -1;

//如有孤立的点一定无法构成最小生成树。

int tot=0;

mset(vis,-1);

mset(idx,-1);

in[root]=0;

rep(i,0,totn){

ans+=in[i];

y=i;

while(vis[y]!=i&&idx[y]==-1&&y!=root){

vis[y]=i;

y=pre[y];

}

//找环。

while(y!=root&&idx[y]==-1){

for(x=pre[y];x!=y;x=pre[x]) idx[x]=tot;

idx[y]=tot++;

}

//对环进行收缩。

}

if(!tot) break;

//如果没有找到新的缩点就结束算法。

rep(i,0,totn) if(idx[i]==-1) idx[i]=tot++;

rep(i,0,totm-1){

int y=ed[i].to;

ed[i].from=idx[ed[i].from];

ed[i].to =idx[ed[i].to];

if(ed[i].from!=ed[i].to) ed[i].val-=in[y];

}

//新建图。

totn=tot-1;

root=idx[root];

//保持根不变

}

return ans;

}

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<cmath>

using namespace std;

const int inf=0x3f3f3f3f;

int vis[105],idx[105],pre[105],cnt;

int in[105],x[105],y[105];

struct node{

int u,v;

int w;

}edge[10005];

int solve(int t,int n,int m){

int x,y,i,j;

int ans=0;

while(1){

for(i=1;i<=n;i++)in[i]=inf;

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

node e=edge[i];

if(e.u!=e.v&&e.w<in[e.v]){

in[e.v]=e.w;

pre[e.v]=e.u;

}

}

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

if(in[i]==inf&&i!=t)return -1;

if(i!=t)ans+=in[i];

}

int tot=0;

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

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

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

y=i;

while(vis[y]!=i&&idx[y]==-1&&y!=t){

vis[y]=i;

y=pre[y];

}

if(y!=t&&idx[y]==-1){

tot++;

idx[y]=tot;

for(x=pre[y];x!=y;x=pre[x])idx[x]=tot;

}

}

if(!tot)break;

for(i=1;i<=n;i++)if(idx[i]==-1)idx[i]=++tot;

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

int y=edge[i].v;

edge[i].u=idx[edge[i].u];

edge[i].v=idx[edge[i].v];

if(edge[i].u!=edge[i].v)edge[i].w-=in[y];

}

n=tot;

t=idx[t];

}

return ans;

}

int main(){

int n,m;

while(scanf("%d%d",&n,&m)!=EOF){

int i,j;

cnt=0;

for(i=1;i<=n;i++)scanf("%lf%lf",&x[i],&y[i]);

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

int f,t;

scanf("%d%d",&f,&t);

if(f==t){

m--;

i--;

continue;

}

double dis;

edge[i].u=f;

edge[i].v=t;

edge[i].w=dis;

}

int ans=solve(1,n,m);

if(ans<0)printf("poor snoopy\n");

else printf("%d\n",ans);

}

return 0;

}