

拓展域并查集

算法简介

一种用来处理各种关系的并查集。

算法例题

[洛谷p1525](#)

题意

给定 \$n\$ 个点 \$m\$ 条带权边。要求将所有点划分为两个点集，输出所有两端点都在同一集合的边权的最大值的最小可能值。

题解

考虑贪心，将所有边按边权从大到小排列，依次加入每条边直到图变成非二分图，答案即为最后无法加入的边的边权。

于是问题转化为如何快速判断当前图是否为二分图，考虑在加边过程中动态维护。

记两个点集为 \$\text{A}, \text{B}\$ 考虑维护 \$2n\$ 条语句，第 \$i\$ 条语句表示 \$v \in \text{A}\$ 第 \$i+n\$ 条语句表示 \$v \in \text{B} (\$ \$1 \leq i \leq n\$)

设当前加的边为 \$u \rightarrow v\$ 于是第 \$u\$ 条语句向第 \$v+n\$ 条语句连一条边，表示 \$u \in \text{A}\$ 当且仅当 \$v \in \text{B}\$ 同理连边 \$v+n \rightarrow u\$

发现“当且仅当”这个关系具有传递性，于是考虑并查集维护。

无法加入边 \$u \rightarrow v\$ 则等价于第 \$u\$ 条语句向第 \$v\$ 条语句属于同一集合，即 \$u \in \text{A}\$ 当且仅当 \$v \in \text{A}\$

```
const int MAXN=2e4+5,MAXM=1e5+5;
struct Edge{
    int u,v,w;
    bool operator < (const Edge &b) const{
        return w>b.w;
    }
}edge[MAXM];
int fa[MAXN<<1];
int Find(int x){
    return x==fa[x]?x:fa[x]=Find(fa[x]);
}
int main()
```

```
{  
    int n=read_int(), m=read_int();  
    _rep(i, 1, n<<1) fa[i]=i;  
    _for(i, 0, m)  
        edge[i].u=read_int(), edge[i].v=read_int(), edge[i].w=read_int();  
        sort(edge, edge+m);  
    _for(i, 0, m){  
        int x=Find(edge[i].u), y=Find(edge[i].v);  
        if(x==y){  
            enter(edge[i].w);  
            return 0;  
        }  
        fa[x]=Find(edge[i].v+n), fa[y]=Find(edge[i].u+n);  
    }  
    enter(0);  
    return 0;  
}
```

算法练习

洛谷p2024

题意

有三种动物 \$A,B,C\$ 仅有 \$A\$ 吃 \$B\$ \$B\$ 吃 \$C\$ \$C\$ 吃 \$A\$ 其他捕食关系均与事实矛盾。

现在有 \$n\$ 只动物，每只动物都属于 \$A,B,C\$ 中的一种。

接下来 \$m\$ 条语句，语句分两类：

1. \$X\$ 与 \$Y\$ 为同类
2. \$X\$ 吃 \$Y\$

一条语句为假当且仅当该语句与之前的真语句矛盾或者与事实矛盾，问共有多少条假语句。

题解

类似的，定义第 \$i\$ 条语句为 \$i \in A\$ 第 \$i+n\$ 条语句为 \$i \in B\$ 第 \$i+2n\$ 条语句为 \$i \in C\$

如果加入的语句为 \$X\$ 与 \$Y\$ 为同类，则连边 \$X \rightarrow Y, X+n \rightarrow Y+n, X+2n \rightarrow Y+2n\$

该语句矛盾等价于 \$(X, Y+n)\$ 或 \$(X+n, Y)\$ 属于同一集合。

如果加入的语句为 \$X\$ 吃 \$Y\$ 则连边 \$X \rightarrow Y+n, X+n \rightarrow Y+2n, X+2n \rightarrow Y\$

该语句矛盾等价于 \$(X, Y)\$ 或 \$(X+n, Y)\$ 属于同一集合。

```
const int MAXN=5e4+5;
int fa[MAXN*3];
int Find(int x){
    return x==fa[x]?x:fa[x]=Find(fa[x]);
}
int main()
{
    int n=read_int(),m=read_int(),opt,u,v,ans=0;
    _rep(i,1,n*3)fa[i]=i;
    _for(i,0,m){
        opt=read_int(),u=read_int(),v=read_int();
        if(u>n||v>n){
            ans++;
            continue;
        }
        if(opt==1){
            if(Find(u)==Find(v+n)||Find(u+n)==Find(v))ans++;
            else{
                fa[Find(u)]=Find(v);
                fa[Find(u+n)]=Find(v+n);
                fa[Find(u+n*2)]=Find(v+n*2);
            }
        }
        else{
            if(Find(u)==Find(v)||Find(u+n)==Find(v))ans++;
            else{
                fa[Find(u)]=Find(v+n);
                fa[Find(u+n)]=Find(v+n*2);
                fa[Find(u+n*2)]=Find(v);
            }
        }
    }
    enter(ans);
    return 0;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E6%8B%93%E5%B1%95%E5%9F%9F%E5%B9%B6%E6%9F%A5%E9%9B%86&rev=1597657931

Last update: 2020/08/17 17:52

