> 计算学部寒假十大打卡活动——“龙舞编程新春会”编程打卡
>
> 2024年1月26日
>
> 洛谷P1525关押罪犯

一、题目

题目描述

S 城现有两座监狱，一共关押着 *N* 名罪犯，编号分别为 1~*N*。他们之间的关系自然也极不和谐。很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。我们用“怨气值”（一个正整数值）来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 *c* 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并造成影响力为 *c* 的冲突事件。

每年年末，警察局会将本年内监狱中的所有冲突事件按影响力从大到小排成一个列表，然后上报到 S 城 Z 市长那里。公务繁忙的 Z 市长只会去看列表中的第一个事件的影响力，如果影响很坏，他就会考虑撤换警察局长。

在详细考察了 *N* 名罪犯间的矛盾关系后，警察局长觉得压力巨大。他准备将罪犯们在两座监狱内重新分配，以求产生的冲突事件影响力都较小，从而保住自己的乌纱帽。假设只要处于同一监狱内的某两个罪犯间有仇恨，那么他们一定会在每年的某个时候发生摩擦。

那么，应如何分配罪犯，才能使 Z 市长看到的那个冲突事件的影响力最小？这个最小值是多少？

输入格式

每行中两个数之间用一个空格隔开。第一行为两个正整数 *N*,*M*，分别表示罪犯的数目以及存在仇恨的罪犯对数。接下来的 *M* 行每行为三个正整数 a_j, b_j, c_j，表示 a_j号和 b_j号罪犯之间存在仇恨，其怨气值为 c_j。数据保证 $1 < a_j \leq b_j \leq N, 0 < c_j \leq 10^9$，且每对罪犯组合只出现一次。

输出格式

共一行，为 Z 市长看到的那个冲突事件的影响力。如果本年内监狱中未发生任何冲突事件，请输出 `0`。

样例 1：

输入：

```
4 6
1 4 2534
2 3 3512
1 2 28351
1 3 6618
2 4 1805
3 4 12884
```

输出

```
3512
```

数据范围

$N \leq 20000, M \leq 100000$

二、解答

思路1：二分答案+二分图

虽然这道题大家一般当作并查集板子题去做，但一拿到题肯定更倾向于二分答案+二分图。

由于是求最小的最大，所以一眼二分枚举最大的影响力，大于二分值的影响力罪犯组合必须分开。

这显然就是个二分图的模板题啦，将需要分开的人之间连边，判断能否将形成的图拆成两边。这个直接用dfs染色法来进行判定就可以了。

二分图染色法的基本原理

1. **两种颜色**：由于二分图的每条边都连接两个不同集合中的节点，因此只需要两种颜色就可以确保没有两个相邻的节点拥有相同的颜色。这是二分图的一个重要性质。
2. **染色步骤**：
   - 首先选择一个集合中的所有节点，给它们染上第一种颜色。
   - 然后，给第二个集合中的所有节点染上第二种颜色。
   - 由于不存在同一集合内的相邻节点，因此这种染色方式保证了所有相邻节点都有不同的颜色。
3. **检测二分性**：此染色法还可以用来检测一个图是否是二分图。如果在尝试使用两种颜色对图进行染色的过程中遇到无法避免的颜色冲突，则说明该图不是二分图。

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=20000+5;
const int maxm=100000+5;

int n,m,mid;

int x[2*maxm],y[2*maxm],z[2*maxm],size=0;
int first[maxn],next[2*maxm],color[maxn];
bool keyi;

void insert(int a,int b,int c)//加边
{
    size++;
    x[size]=a;
    y[size]=b;
    z[size]=c;
    next[size]=first[a];
    first[a]=size;
}

void readdata()
{
    memset(first,-1,sizeof(first));
    memset(next,-1,sizeof(next));

    scanf("%d%d",&n,&m);
    for (int i=1;i<=m;i++)
    {
```

```
        int a,b,c;
        scanf("%d%d%d",&a,&b,&c);
        insert(a,b,c);
        insert(b,a,c);
    }
}

void ran(int m1,int xx,int c)//染色法判定是否存在矛盾
{
    if (!keyi) return;
    if(color[xx]!=-1)//已经染过色
    {
        if(color[xx]!=c)//染色矛盾
        {
            keyi=false;
            return;
        }
        else return;//记搜提前返回
    }

    color[xx]=c;//染色
    for (int e=first[xx];e!=-1;e=next[e])
    {
        int a,b,cc;
        a=x[e];
        b=y[e];
        cc=z[e];
        if(cc>m1) //权值大于答案值
        ran(m1,b,1-c);
     }
}

bool check(int mm)
{
    keyi=true;
    memset(color,-1,sizeof(color));
    for (int i=1;i<=n;i++)
    {
        if (-1==color[i])
        {
            ran(mm,i,1);
        }
    }
    return keyi;
}

void work()//二分答案
{
    int l=0,r=1000000000;
    while(l<r)
    {
        int mid=(l+r)/2;
        if(check(mid)) r=mid;
        else l=mid+1;
    }
```

```
    printf("%d",l);
}

int main()
{
    readdata();
    work();
    return 0;
}
```

思路2：并查集

我甚至第二种解法一开始也是二分答案+并查集去做的，写着写着突然发现没必要，直接将所有边按权值从大到小排序后按顺序进行并查集算法，到第一次发现要进行合并的两条边已经在同一个集合中就停止即可。

并查集：因为是使满足条件边的两点分开，不如设置点i的绝对敌人点i+n，将其与i的所有敌人（与罪犯i怨气值大于答案值的人）合并入一个集合。从而将与i不在同一个集合（监狱）问题转换成与i+n在同一个集合中。显然对于同一个集合的罪犯来说，由于错综复杂的关系，他们必须关押在同一个集合（监狱）中。因此，按照从大到小的权值加边，如果边的两个端点不在同一个集合中，他们可以与对方绝对敌人点并入同一个集合中，否则说明两个人必须在同一个集合中，此时的边权值val就是答案所求的最大怨气值。

```
#include<bits/stdc++.h>
using namespace std;

const int maxn=20005,maxm=100005;
int n,m,fa[maxn<<1];
struct edge{
    int x,y,val;
    bool operator<(const edge &aa)const{//结构体重载运算符，方便直接用sort排序
        return val>aa.val;//大的val值优先级更高
    }
}e[maxm];

int find(int x){
    return fa[x]==x?x:fa[x]=find(fa[x]);
}

int main()
{
//  freopen("in.txt","r",stdin);
    scanf("%d %d",&n,&m);
    for(int i=1;i<=m;i++)
        scanf("%d %d %d",&e[i].x,&e[i].y,&e[i].val);

    sort(e+1,e+m+1);


    for(int i=1;i<=n+n;i++)fa[i]=i;
    for(int i=1;i<=m;i++)
    {
        int x=e[i].x,y=e[i].y;
```

```
        if(find(fa[x])==find(fa[y])){
            printf("%d ",e[i].val);
            return 0;
        }
        fa[find(x)]=find(y+n);
        fa[find(y)]=find(x+n);
    }
    printf("0");
    return 0;
}
```