边的染色

一. 考察内容：

图论 图上计数问题

二. 题目分析：

[题目大意]

在一个有向图中，有一些给定权值的边和一些不定权值的边，求有多少种对不定权值的边赋权的方案，使得图让任意一个环上边的异或值都为0。

[写题思路]

先将所有边都当做没有初始值的边，我们随意的给图的每个节点赋上一些权值（1/0），对于一条边，他的权值设为其两个端点权值的异或值，这样，对于每个环，所有边的异或值都为0，因为相当于每个节点都被异或了两遍。

现在只考虑那些确定权值的边，对于每一个连通块，我们随意的对一个点赋权，然后通过dfs的方式为其他点赋权，要满足任意一条边两端点异或值，如果可以满足，则有解，否则无解。

最后，将原图建出来，对于每个连通块，有多少种方案呢？我们可以求出赋权的方案数，如果有n个点，没有给定权值的边，那么方案数为2^n，但是由于把每个点的权值都反过来之后边权依然不变，所以方案数就为2^(n-1)，每添加一条确定权值的边，就相当于固定了两个点之间的权值关系，用并查集来维护，如果这两个点原本不属于一个集合，那么方案数就会除以2，如果这两个点原本属于一个集合，那么方案数不变，最后求出方案数即可。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/*************************
*创建时间：2018 08 16
*文件类型：源代码文件
*题目来源：牛客
*当前状态：已通过
*备忘录：图论 无向图
*作者：HtBest
*************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
using namespace std;
namespace IO
{
    #define SIZE 1
    int T=0,end=0;
    char s[SIZE+1];
    int in()
    {
        T=0;
        return end=fread(s,1,SIZE,stdin);
    }
    int read(int &a)
    {
```

```
        int i=a=0,flag=1;
        while(1)
        {
            if(T==end&&!in())return i;
            if(s[T]==' '||s[T]=='\n'||s[T]=='\r')
            {
                if(i)return i;
            }
            else if(s[T]=='-')flag=-1;
            else
            {
                a=flag*(flag*a*10+s[T]-'0');
                ++i;
            }
            ++T;
        }
    }
}
#define MAXN 100002
#define MOD 998244353
int n,m,head[MAXN],_edge,dfsx[MAXN],_dfsx,v[MAXN],ans=1,noden ,edgen;
struct EDGE
{
    int a,b,v,next;
    EDGE(int a=0,int b=0,int v=0,int next=0):a(a),b(b),v(v),next(next){}
}edge[2*MAXN],a[MAXN];
struct UFS
{
    int f[MAXN];
    int find(int x)
    {
        return x==f[x]?x:f[x]=find(f[x]);
    }
    void merge(int a,int b)
    {
        f[find(a)]=find(b);
    }
    void set()
    {
        for(int i=1;i<=n;++i)f[i]=i;
    }
}ufs;
/* Variable explain:

*/
int add(int a,int b)
{
    return (a+b)%MOD;
}
int mul(int a,int b)
{
    return (long long)a*b%MOD;
}
void adde(int a,int b,int v)
{
    edge[++_edge]=EDGE(a,b,v,head[a]);
    head[a]=_edge;
}
void adde(EDGE a)
{
    adde(a.a,a.b,a.v),adde(a.b,a.a,a.v);
}
void read()
{
    using IO::read;
    // freopen(".in","r",stdin);
```

```
    // freopen(".out","w",stdout);
    read(n);
    read(m);
    for(int i=1;i<=m;++i)
    {
        read(a[i].a);
        read(a[i].b);
        read(a[i].v);
        if(a[i].v!=-1)  adde(a[i]);
    }
    return;
}
bool dfs1(int a)
{
    dfsx[a]=++_dfsx;
    for(int i=head[a];i;i=edge[i].next)
    {
        int b=edge[i].b;
        if(dfsx[b])
        {
            if((v[a]^v[b])!=edge[i].v)return false;
        }
        else
        {
            v[b]=v[a]^edge[i].v;
            if(!dfs1(b))return false;
        }
    }
    return true;
}
void dfs2(int a,int l)
{
    ++noden;
    dfsx[a]=++_dfsx;
    for(int i=head[a];i;i=edge[i].next)
    {
        int b=edge[i].b;
        if(l==(i^1))continue;
        if(edge[i].v!=-1)
        {
            if(ufs.find(a)!=ufs.find(b))
            {
                ufs.merge(a,b);
                ++edgen;
            }
        }
        if(!dfsx[b])dfs2(b,i);
    }
}
int pow(int a,int n)
{
    int ans=1;
    while(n)
    {
        if(n&1)ans=mul(a,ans);
        a=mul(a,a);
        n>>=1;
    }
    return ans;
}
int main()
{
    read();
    for(int i=1;i<=n;++i)
        if(!dfsx[i])
            if(!dfs1(i))
```

```
            {
                printf("0\n");
                // for(int i=1;i<=_edge;++i)printf("%d ",v[i]);
                exit(0);
            }
    for(int i=1;i<=n;++i)dfsx[i]=head[i]=0;
    _dfsx=0,_edge=1;
    ufs.set();
    for(int i=1;i<=m;++i)adde(a[i]);
    for(int i=1;i<=n;++i)
    {
        if(!dfsx[i])
        {
            noden=edgen=0;
            dfs2(i,0);
            // printf("%d %d\n",noden,edgen);
            ans=mul(ans,pow(2,noden-edgen-1));
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

<题目跳转> <查看代码>