spfa应用之成环判定

闲聊

前天夜里做第届省赛的题信道安全的时候发现这个题用每一次都找最大的。这就尴尬了记得我上一篇博客写的是求最短的。遇到这一题我开始有一点懵啊到夜里回寝室也想出来，试过了修改狄克斯特拉和prim都失败了(虽然我感觉也可以)最后我发现用spfa就是一个套路了然后那一题就水过了。今天这一题又不一样了不止每一次找最大的还用到了spfa的判定成环。。

题目信息

描述

AC_Grazy一直对江湖羡慕不已,向往着大碗吃肉大碗喝酒的豪情，但是“人在江湖漂,怎能不挨刀",”人在江湖身不由己",如果自己的武功太差,在江湖会死的很惨，但是AC_Grazy没有武功秘籍练不了绝世武功.有道是“山重水复疑无路,柳暗花明又一村”,在AC_Grazy家里面竟然藏着一本书,书名竟然叫做【超级外挂】,竟然能在各种武功之间进行转化,据说是他爷爷的爷爷的...爷爷传下来的...

闲着无事便拿来看看,只看一眼便再也停不下了,只见上面写着“纵横武林打遍天下无敌手武功心法秘籍收录”.

翻开第一篇一看竟然是【降龙十八掌】...

心法只是一个修练武功的途径,重要的是真气的多少,于是他便想利用外挂让武功之间进行转化，来让真气无限增加，但是这个心法只能按照顺序转化，我们分别用 1号和2号来代替两种功法 当然转化会有一定的转化率f

比如1 0.5 2 便是把 1的一半真气转化给2 ,为了简化问题，我们每次都从1号秘籍开始进行转化,如果其中一个秘籍转化断了，那么以后的功法就不能转换。

输入

输入：首先输入一个数 T(T<=20)表示T组数据

然后输入两个数n(2<=n<=500)和m(1=<m<=2000)分别表

示有n种秘籍，随后的m行分别输入

秘籍u(n>=u>0) 转化率 f (0<f<=10)秘籍 v.(0<v<=n)

输出

输出：如果可以无限增加真气输出Yes否则输出No.

样例输入

```
2
3 3
1 2 2
2 2 3
3 2 1
4 3
1 2 2
3 2 4
4 2 3
```

样例输出

```
Yes
No
```

AC代码

```
#include <iostream>
#include <vector>
#include <queue>
#include <string.h>
#include <stdio.h>

#define INF 0x3f3f3f3f
using namespace std;

struct node{
int y;
double quan;

}t;


double d[50005];
int xx[50005];//判断节点是否在队中
int chi[50005];//表示这个节点被访问的次数
int in_degree[5005];//节点的入度
int main()
{
    int n;
    int l;
    int T;
    bool ff=false;
    scanf("%d",&T);
    while(T--)
    {
    vector<node>mymap[10005];
    queue<int>Q;
    scanf("%d%d",&n,&l);
    memset(d,0xc0c0c0c0,sizeof(d));
    memset(xx,INF,sizeof(xx));
    memset(chi,0,sizeof(chi));
    memset(in_degree,0,sizeof(in_degree));
    for(int i=1;i<=l;i++)
    {
        int a,b;
        double c;
        cin>>a>>c>>b;
        t.quan=c;
        t.y=b;
        mymap[a].push_back(t);
        in_degree[b]++;
    //  t.y=a;
    // mymap[b].push_back(t);
    }
    Q.push(1);
    d[1]=1.0;
    xx[1]=0;
    chi[1]++;
    while(!Q.empty())
    {
        int tmp=Q.front();
        Q.pop();
        xx[tmp]=INF;
        for(unsigned int i=0;i<mymap[tmp].size();i++)
        {
        if(mymap[tmp][i].quan*d[tmp]>d[mymap[tmp][i].y]) //
```

```
        {
            d[mymap[tmp][i].y]=mymap[tmp][i].quan*d[tmp];
            if(xx[mymap[tmp][i].y]!=0)
            {
                Q.push(mymap[tmp][i].y);
                xx[mymap[tmp][i].y]=0;
                if(++chi[mymap[tmp][i].y]>in_degree[mymap[tmp][i].y])
                {
                    ff=true;////已经成环  真气可以无限增加
                    break;
                }

            }
        }

        }
    }

    if(ff)
    {
        cout<<"Yes"<<endl;
        ff=false;
    }
    else
        cout<<"No"<<endl;
    }
    return 0;
}
```

作者语

这个代码是我从上一篇博客修改而来的主要想法