title: 并查集判环与spfa寻找最长的路径

并查集判环与spfa寻找最长的路径

题目描述

湫湫系列故事——设计风景线

Problem Description

随着杭州西湖的知名度的进一步提升，园林规划专家湫湫希望设计出一条新的经典观光线路，根据老板马小腾的指示，新的风景线最好能建成环形，如果没有条件建成环形，那就建的越长越好。

现在已经勘探确定了 n 个位置可以用来建设，在它们之间也勘探确定了 m 条可以设计的路线以及他们的长度。请问是否能够建成环形的风景线？如果不能，风景线最长能够达到多少？

- 其中，可以兴建的路线均是双向的，他们之间的长度均大于 0。

Input

测试数据有多组，每组测试数据的第一行有两个数字 n, m，其含义参见题目描述；

接下去 m 行，每行 3 个数字 u v w，分别代表这条线路的起点，终点和长度。

[Technical Specification]

1. n<=100000
2. m <= 1000000
3. 1<= u, v <= n
4. w <= 1000

Output

对于每组测试数据，如果能够建成环形（并不需要连接上去全部的风景点），那么输出 YES，否则输出最长的长度，每组数据输出一行。

Sample Input

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

Sample Output

```
YES
```

解题思路

从题目中我们可以发现首先我们要判定输入的数据是不是可以构成环。我们通过 find(a) 和find(b) 来找到啊 a,b 两点的根节点，如果 fa == fb 了表示两点在一个集合里又由于a 和 b 中间有一条路所以这个集合已经成环了。 如果fa != fb了 我们要将以fa与fb为根的两个集合合并为一个join(fa, fb)

然后开始用add(a,b) 与 add(b,a) 建立一个双向图 直到我们的数据输入完毕

如果我们在输入的时候已经发现有联通图了我们直接输出 “YES” 就可以了

如果没有联通的话我们就要通过搜索寻找最长的简单路

由于我们建立的邻接表里面是保存的森林我们要遍历每一个根节点。

通过SPFA算法先找到距离跟节点最长的子叶节点，然后通过这个子叶节点寻找距离这个节点最长的节点

通过以上的步骤就可以找到我们想要的最长的简单路了

AC代码

```
#include<stdio.h>
#include<string.h>
#include<queue>
#include<algorithm>
#define MAX 100010
using namespace std;
int pre[MAX],ran[MAX],flag;
struct node
{
    int to,val,next;
}edge[MAX*10];
int head[MAX],dis[MAX];
bool vis[MAX];
int top,T,ans;
void init(int n)
{
    ans=-1;
    top=0;
    int i;
    for(i=0;i<=n;i++)
    pre[i]=i;
    memset(ran,0,sizeof(ran));
    memset(head,-1,sizeof(head));
}
int find(int x)
{
    if(x==pre[x])
    return x;
    return pre[x]=find(pre[x]);
}
void join(int x,int y)//加权规则，判断环
{
    if(x==y)
    return ;
    if(ran[x]>=ran[y])
    pre[y]=x;
    else
    {
    if(ran[x]==ran[y])
    ran[y]++;
    pre[x]=y;
    }
}
void add(int u,int v,int w)
{
    //edge[top].from=u;
    edge[top].to=v;
    edge[top].val=w;
    edge[top].next=head[u];
    head[u]=top++;
}
void SPFA(int s)
```

```
{
    memset(vis,false,sizeof(vis));
    memset(dis,0,sizeof(dis));
    queue<int>q;
    dis[s]=0;
    vis[s]=true;
    q.push(s);
    T=s;
    ans=0;
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        for(int i=head[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].to;
            if(!vis[v])
            {
                if(dis[v]<dis[u]+edge[i].val)
                {
                    dis[v]=dis[u]+edge[i].val;
                    if(ans<dis[v])
                    {
                        ans=dis[v];
                        T=v;
                    }
                }
                vis[v]=true;
                q.push(v);
            }
        }
    }
}
int main()
{
    int n,m,a,b,c,i;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        bool flag=false;
        init(n);
        for(i=0;i<m;i++)
        {
            scanf("%d%d%d",&a,&b,&c);
            if(!flag)//还不存在环，仍需要判断,否则输入完了不用管
        {
                int fa=find(a);
                int fb=find(b);
                if(fa==fb)//存在环了
                {
                    flag=true;
                }
                else
                {
                join(fa,fb);
                add(a,b,c);
                add(b,a,c);
                }
            }
        }
        if(flag)
        {
```

```
            printf("YES\n");
        }
        else
        {
            int num=0;
            for(i=1;i<=n;i++)
            {
                if(pre[i]==i&&head[i]!=-1)//一个树，并且有子树，不是一个点
                {
                    SPFA(i);//寻找最远的子叶节点
                    SPFA(T);//从最远的子叶节点寻找最最长的路径
                    num=max(num,ans);
                }
            }
            printf("%d\n",num);
        }
    }
    return 0;
}
```