I题题解：2-sat问题

题意：n个节点，2种状态，m个约束条件，每个约束条件需至少满足两个子约束条件中的一个。
问是否存在一组n使得题目成立。存在则输出YES，否则输出NO。

先拿样例理解下题目。

input	output
3 2 1 1 3 1 2 0 3 0	YES

节点1为1或者节点3为1；
节点2为0或者节点3为0；
对于这样小的数据，我们很容易找到满足题目条件的一组ai；
比如说，a1=1,a2=0,a3=1;a1=0,a2=0,a3=1;
于是我们就输出YES。
但是如果数据量很大该怎么办？
观察下输入，我们可以联想到离散数学中学到的一个公式：（其中P和Q均为命题）

> PvQ=非P->Q^非Q->P（不好意思我的markdown貌似功能有些缺失没法打数学符号）

在该题目中，假设Pi为节点i是1，那么非Pi则为节点i是0；

由以上，样例可以理解为：

- 如果(节点）1是0，那么3一定是1；
- 如果3是0, 那么1一定是1；
- 如果2是1，那么3一定是0；
- 如果3是1，那么2一定是0；

我们发现，m个约束条件可以推出2*m个带有方向的约束条件。
假如说，我们把每个节点分成两个节点，分别代表其两种状态。
比如说，a1,a2,a3分别表示a1是1，a2是1，a3是1；
a4,a5,a6分别表示a1是0，a2是0，a3是0；
即，对于原第i个节点，ai表示它的第一个状态，a[i+n]表示它的第二个状态。

```
//数据的预处理
//这里采用前向星存图
for(int i=0;i<m;++i) {
    int x,y,xVal,yVal;
    scanf("%d%d%d%d",&x,&xVal,&y,&yVal);
  int tag1=1-xVal,tag2=1-yVal;
  add(x+tag1*n,y+yVal*n);
  add(y+tag2*n,x+xVal*n);
}//不理解tag1、tag2的话可以自己模拟下
```

那么，输出NO的情况是怎样的？
其实就是，某个节点同时拥有两种状态。或者说，它不得不拥有两种状态。

说的再具体点，就是，假如我们从建图后形成的节点ai出发（这个意思其实就是假定ai是真的），把所有能走到的节点都走一遍，最后发现它把a[i+n]也走到了。
也就是说，如果ai是真的,那么a[i+n]也必须是真的。矛盾，故不存在。
或者，从ai开始走不会走到a[i+n]，但是ai并没有把所有图走一遍，这时候就要再找另外的节点开始走，直到把整个图走过一遍。
如果在其他图走的时候，同样走到了a[i+n]，同样矛盾。
但是，总不能针对每个ai,a[i+n]，都要独立地搜索一次，这恐怕会很复杂。
这时候就要说一句：

> ## Tarjan大法好

我们知道，Tarjan可以用来求强连通分量。
什么是强连通分量呢？就是说在这个子图中，任意两个节点是可以互达的。
代入到这个题目中，是什么意思呢？
在强连通分量里，只要满足一个子约束条件，其余所有子约束条件都会满足，
无论你从哪个节点开始走! 之前我在学习2—sat问题时就有疑惑，但是理解下，**同一个节点只会出现在一个连通分量里**。
它不会因为你从哪个节点开始走而改变。
于是乎，我们a1循环到a2n，如果ai没有被标记，那么就从它开始走一遍tarjan，保证所有可以有连通分量的节点都有其归属连通分量。
如果某一对ai，a[i+n]被分在了同一个连通分量里，那么很不幸，我们没法找到这样一组ai来满足题意。
所以输出"NO"；
如果任何一对ai,a[i+n]都不在同一个连通分量里，那么就可以输出”YES”了。

OK，下面结合代码总结下算法步骤：

1. 利用逻辑关系的等价建边构图（代码上面有）； 利用tarjan求强连通分量，同时记录每个强连通分量所包含的顶点（注意这里要用到栈了）；

```
void Tarjan(int x) {
    ……
    ……
    if(dfn[x]==low[x]) { //dfn[i]=low[i]，是找到一个连通分量的标志
        num_scc++;
        while(1) {
            int temp=st.top();//出栈
            st.pop();
            vis[temp]=false;
            scc[temp]=num_scc;
            if(temp==x)//一个节点只属于一个连通分量，所以x也出栈
                break;
         }
     }
}
```

2. 第二个步骤从a1到a2n，ai没被标记则tarjan一次（以保证所有可以有归属连通分量的点都被记录到相应的连通分量里）；

```
for(int i=1;i<=2*n;i++)
    if(!dfn[i])
        Tarjan(i);
```

3. 考察每对ai,a[i+n]是否在同一个连通分量里，如果存在这么一对，则输出"NO"；否则输出"YES"。

```
for(int i=1;i<=n;i++){
    if(scc[i]==scc[i+n]){
     printf("NO\n");
     return 0;
 }
}
printf("YES\n");
return 0;
```

> **时间复杂度O（V+E）：**
>
> 所有节点都要判断一遍要不要跑tarjan，tarjan一共只把每个边都跑一边，所以就是O（V+E）。嗯就这样子。