信奥算法

并查集

Union-Find Disjoint Sets

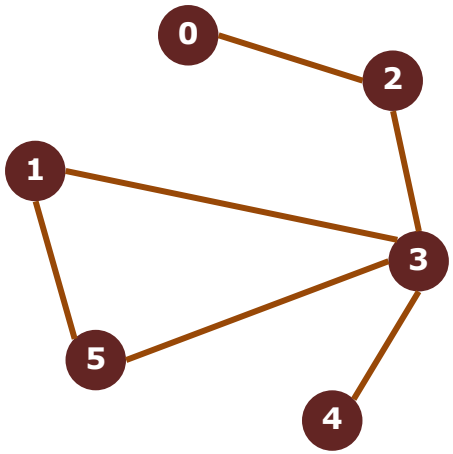
合并+查找

数据结构

攀亲戚

你发现自己和古代一个皇帝长得很像：都有两个眼睛一个鼻子，你想知道皇帝是不是你的远方亲戚，你是不是皇亲国戚。目前你能掌握的信息有m条，关于n个人：第i条信息包含两个人的编号ai,bi，表示ai和bi是亲戚。你的编号是0，皇帝的编号是1，最大编号为n-1，请问能否通过信息推理出你和皇帝是不是亲戚？
备注：亲戚关系具有传递性。输入m和n，均不超过1000。

输入样例：
6 6
0 2
3 4
5 1
2 3
3 5
1 3
输出样例：
Yes



图论建模要素

每个节点代表1个人

每条无向边连接2个节点

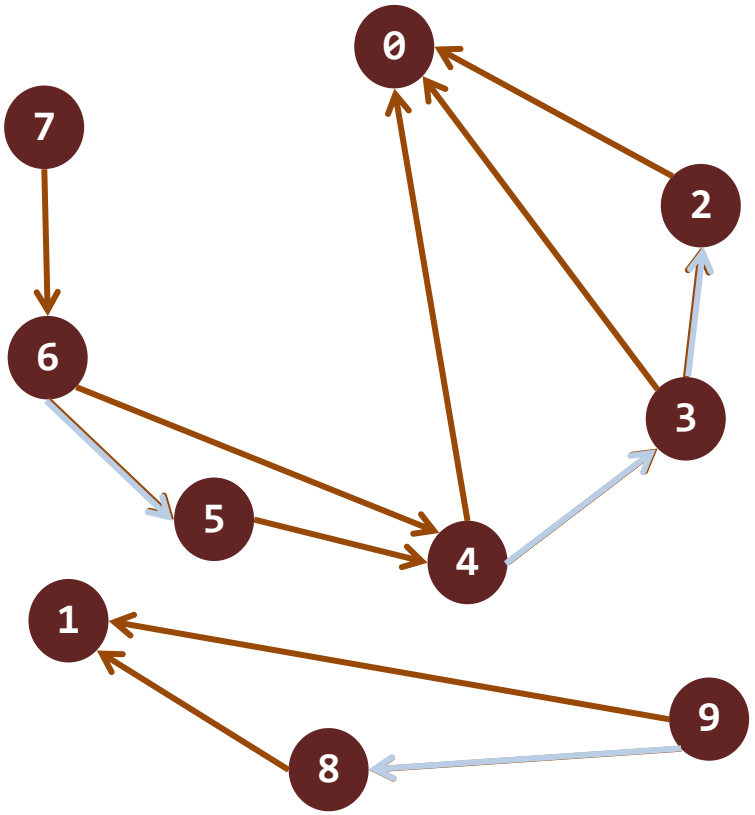
代表这2个人是直接亲戚

问题：0号和1号节点是否连通

id[i]代表i号节点指向几号

i	0	1	2	3	4	5	6	7	8	9
id[i]	0	1	0	0	0	4	4	6	1	1

连接关系	
0	2
4	5
6	7
1	8
8	9
2	3
5	6
3	4



无向边
变有向边

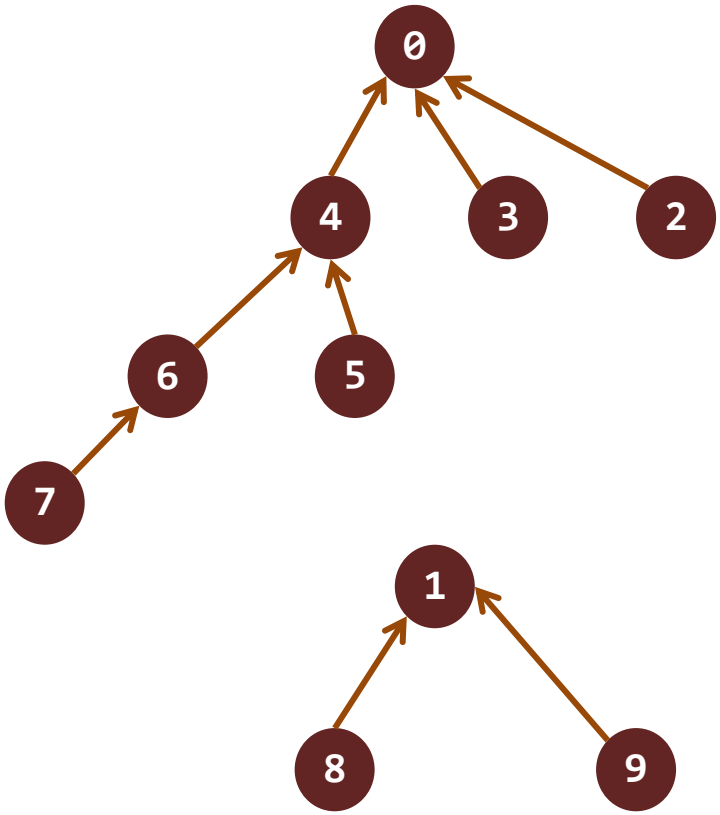
每个连通块
有个根节点

祖先数组

id[i]代表i号节点指向几号

i	0	1	2	3	4	5	6	7	8	9
id[i]	0	1	0	0	0	4	4	6	1	1

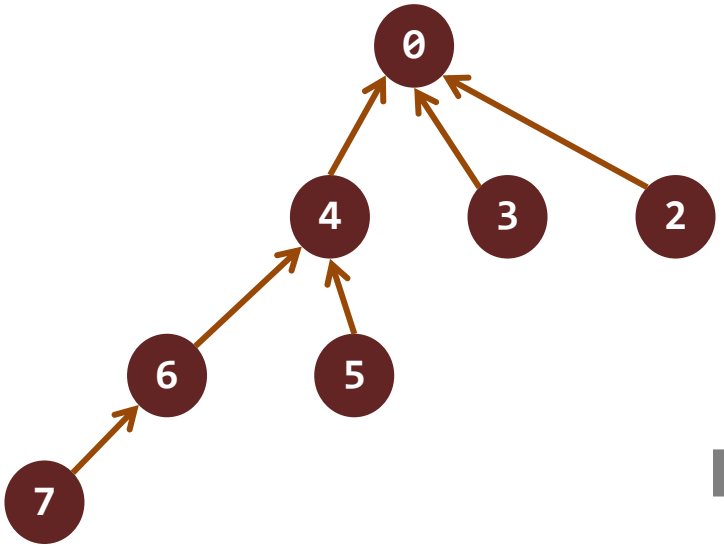
连接关系	
0	2
4	5
6	7
1	8
8	9
2	3
5	6
3	4

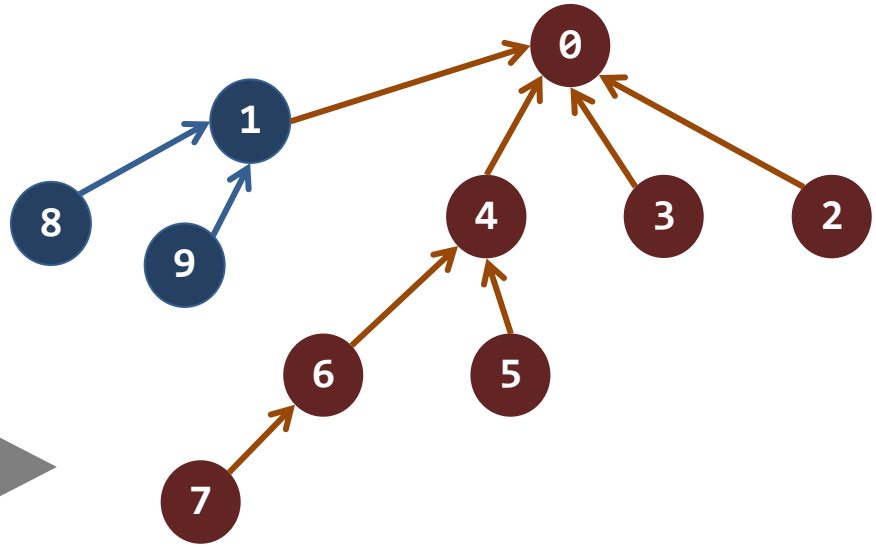


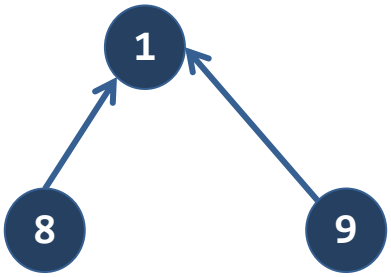
无向边
变有向边

每个连通块
有个根节点

"老祖宗"







正确	id[1]=0;或id[0]=1;
错误	id[7]=8;或id[8]=7;

加一条边连接7和8
7的老祖宗是0
8的老祖宗是1

让1认0做爸爸

攀亲戚：主函数

```
3  typedef long long ll;
4  const ll N=1009;
5  ll n,m,x,y,id[N];
14     cin>>m>>n;
15     for(ll i=0;i<n;i++)id[i]=i;

16     while(m--){
17         cin>>x>>y;
18         unite(x,y);
19     }

20     if(root(1)==root(0))cout<<"Yes"<<endl;
21     else cout<<"No"<<endl;
```

初始化:每个节点i指向自己

易错点

共m条边,每条边端点x,y
合并x和y所在的两个树

若1号所在树的根节点
等于0号所在树的根节点

unite()并　root()查

```
6  ll root(ll x){
7      if(id[x]==x)return x;
8      else return root(id[x]);
9  }
```

```
10 void unite(ll x,ll y){
11     ll rx=root(x),ry=root(y);
12     id[rx]=ry;
13 }
```

易错点	id[x]=y;	区分爸爸和祖宗

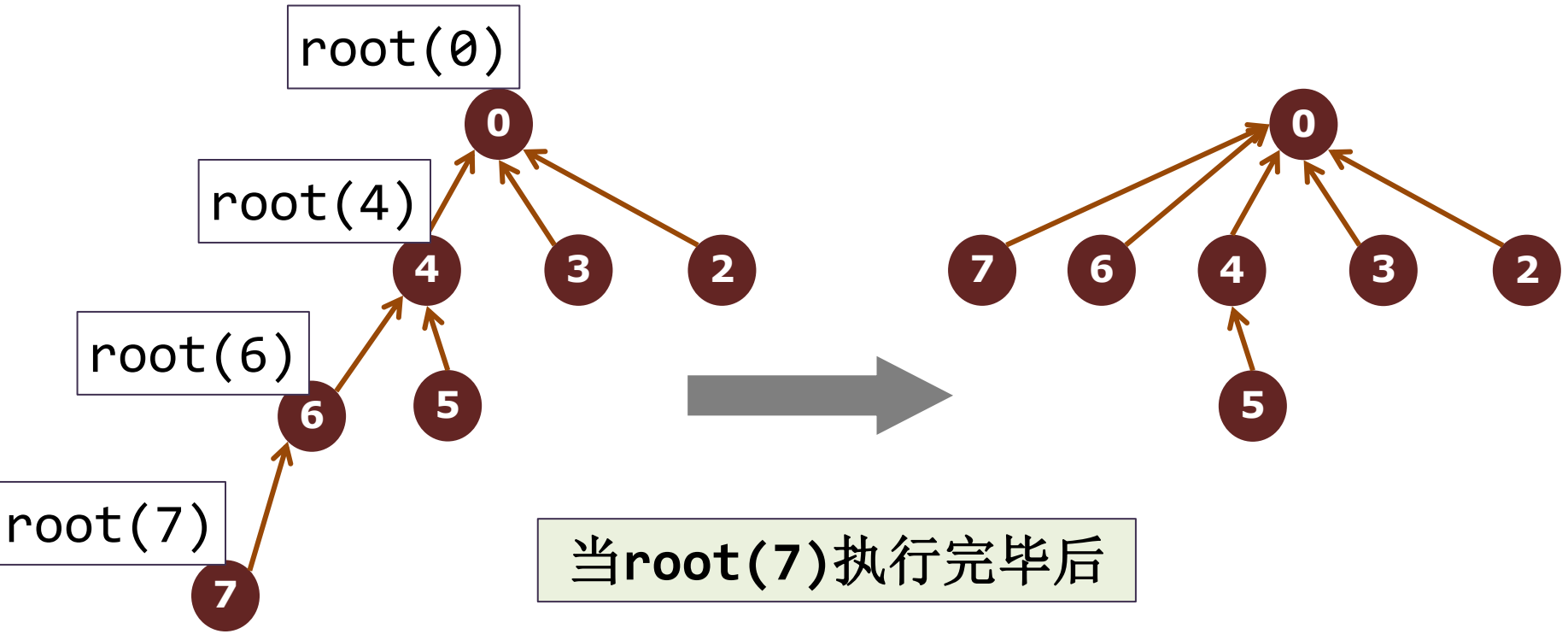
unite()并　root()查

```
6  寻找x的老祖宗编号
7      若x的爸爸就是x自己 返回x
8      否则 继续寻找x爸爸的老祖宗
9  }
```

错误翻译	合并x和y两个节点

```
10 合并x和y所在的两个树
11     rx为x的老祖宗，ry为y的老祖宗
12     让ry成为rx的爸爸
13 }
```

优化：路径压缩



当root(7)执行完毕后

7号节点到0号根节点
路径上途径节点7,6,4
都重新指向0号

路径压缩 root()查

路径压缩两行代码

```
6 ll root(ll x){
7     if(id[x]==x)return x;
8     return id[x]=root(id[x]);
9 }
```

路径压缩一行代码

```
6 ll root(ll x){
7     return id[x]==x?x:id[x]=root(id[x]);
8 }
```

平摊时间复杂度
接近O(1)

连通性 三件套

DFS

BFS

动态修改
不好处理

并查集

支持动态
连边和查询

但原图信息
有所丢失

现场挑战
太戈编程547

547关系式矛盾2：n条关系式，判断是否有可能都成立。关系符号!=代表不等于，==代表等于。变量字母a到z。

1<=n<=100

图论建模要素

每个节点代表什么?	1个变量符号
每条边代表什么?	1条关系式

n是点数还是边数?	边数
26是点数还是边数?	点数

算法步骤

先将等式都合并
形成若干连通块

依次处理不等式

若存在不等式u!=v
但是u,v已经通过若干等式
合并在同一个连通块里

输出Impossible
结束程序

输出possible

```
10      ll nRelations,m=0;
11      cin>>nRelations;
12      for(ll i=0;i<N;i++)id[i]=i;
13      for(ll i=0;i<nRelations;i++){
14          char u,v,x,y;
15          cin>>u>>x>>y>>v;
16          if(x=='=')
17              unite(u-'a',v-'a');
18          else {
19              qx[m]=u-'a';
20              qy[m]=v-'a';
21              m++;
22          }
23      }
```

因为原题中n容易混淆
改成nRelations明确含义

N是几

如果是等式

qx[m]代表m号不等式的左边变量

qy[m]代表m号不等式的右边变量

m是当前不等式编号

m代表不等式总数

qx[i]代表i号不等式的左边变量

qy[i]代表i号不等式的右边变量

```
24      for(int i=0;i<m;i++)
25          if(                              ){
26              cout<<"Impossible"<<endl;
27              return 0;
28          }
29      cout<<"Possible"<<endl;
```

补全程序

现场挑战
太戈编程1730

算法1 模拟+并查集

动态维护

连通块个数

按照题意依次恢复公路
按时间顺序模拟每一步

并查集合并连通块

每次合并2个连通块时
连通块个数减少1

连通块个数为1时
所有城市连通

动态维护

每次合并2个连通块时
连通块个数减少1

```
13      int cnt=m;
14      for(ll t=1;t<=n;t++){
15          int u,v;
16          cin>>u>>v;
17          int ru=root(u);
18          int rv=root(v);
19          [          ]
20          unite(ru,rv);
21          cnt--;
22          if([      ]){
23              cout<<t<<endl;
24              return 0;
25          }
26      }
27      cout<<[ ]<<endl;
```

补全程序

算法2 二分答案+DFS

二分枚举答案：第x天

判断第x天时
所有城市是否连通

并查集演示

算法可视化网址：
visualgo.net/en/ufds

The Union-Find Disjoint Sets (UFDS) data structure is used to model a collection of disjoint sets, which is able to efficiently (i.e. in nearly constant time) determine which set an item belongs to, test if two items belong to the same set, and union two disjoint sets into one when needed. It can be used to find connected components in an undirected graph, and can hence be used as part of Kruskal's algorithm for the Minimum Spanning Tree (MST) problem.

太戈编程

2533,1733

要求	搭配暴力+对拍