3.并查集

Union-Find Set

赵宗昌

- **一类问题：**
- 给出集合*S={1,2,..n}*，和定义在*S*上的一个等价关系集*R*。
- 1.合并两个集合
- *2.判断两个元素u和v是否属于同一个集合。*
- *3.S*中不同等价类的数目(集合)。

引例：

- 某个大的公司集团下有若干个子公司，集团共有n个员工（由于人太多，不同子公司的人相互不认识），现在给定关于 n个人的m条信息（即某2个人认识），假设所有认识(直接或间接)的人一定属于同一个子公司，请计算该集团共有多少个子公司？
- 若是某两人不在给出的信息里，那么他们不认识，属于两个不同子公司。
- 已知人的编号从1至n。
- **输入：**
- 第一行：n（<=10000,人数），
- 第二行：m（<=100000，信息）
- 以下若干行：每行两个数：i 和j，中间一个空格隔开，表示i和j相互认识。
- **输出：**
- 子公司的数量。

样例输入：

11
9
1 2
4 5
3 4
1 3
5 6
7 10
5 10
6 10
8 9

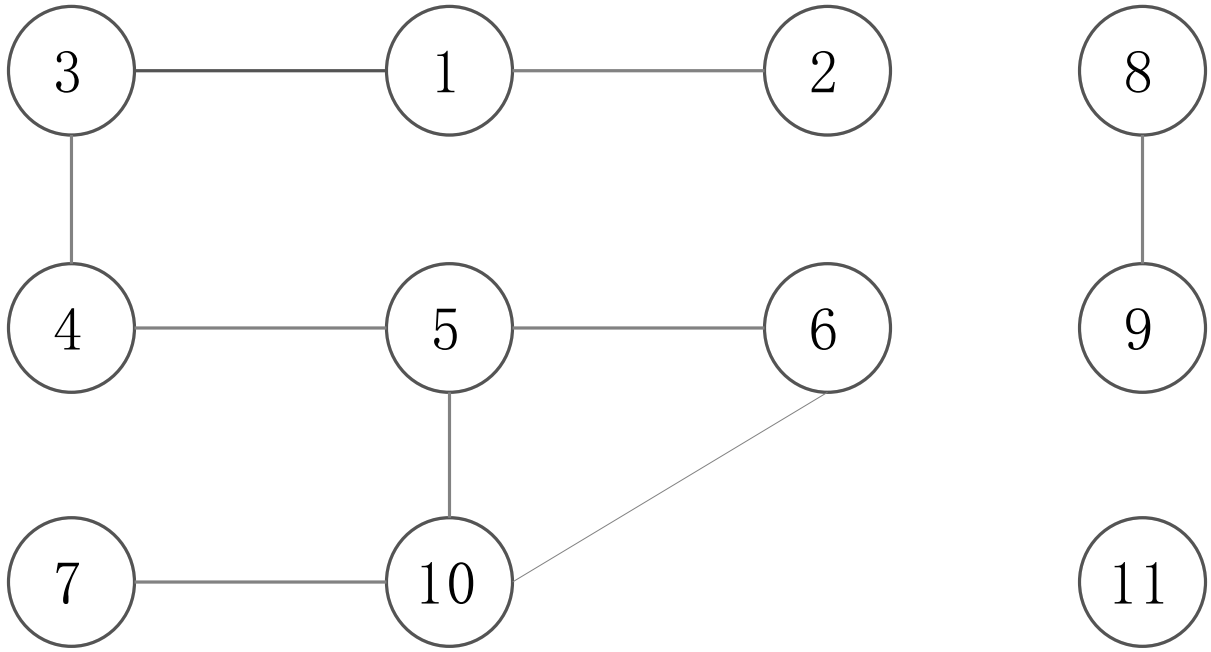
样例输出：

3

算法1：图的连通分量

样例输入：

11
9
1 2
4 5
3 4
1 3
5 6
7 10
5 10
6 10
8 9



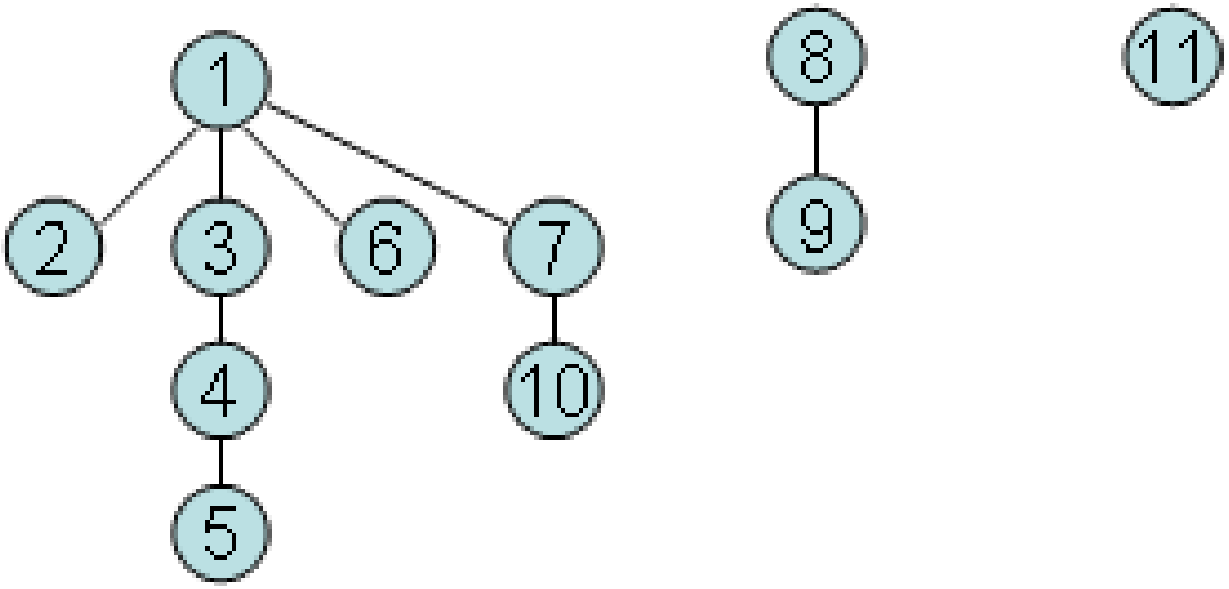
样例输出：

3

算法2：

样例输入：

11
9
1 2
4 5
3 4
1 3
5 6
7 10
5 10
6 10
8 9



样例输出：

3

◆并查集的巧妙之处在于用树来表示集合。用于处理一些不相交集合S={S1，S2，…,Sn}，每个集合Si都有一个特殊元素root[Si]（树根）,称为集合代表.

◆并查集支持三种操作：

Init（X）: 集合初始化：x的父节点p[x]=x;

Find(x): 查找：查找x所在集合的树根。

优化：路径压缩。

Union(x, y): 合并：把x和y所在的两个不同集合合并。

三种操作：

Init（X）：

集合初始化：**fa[x]=x;**每个结点的都是树根.

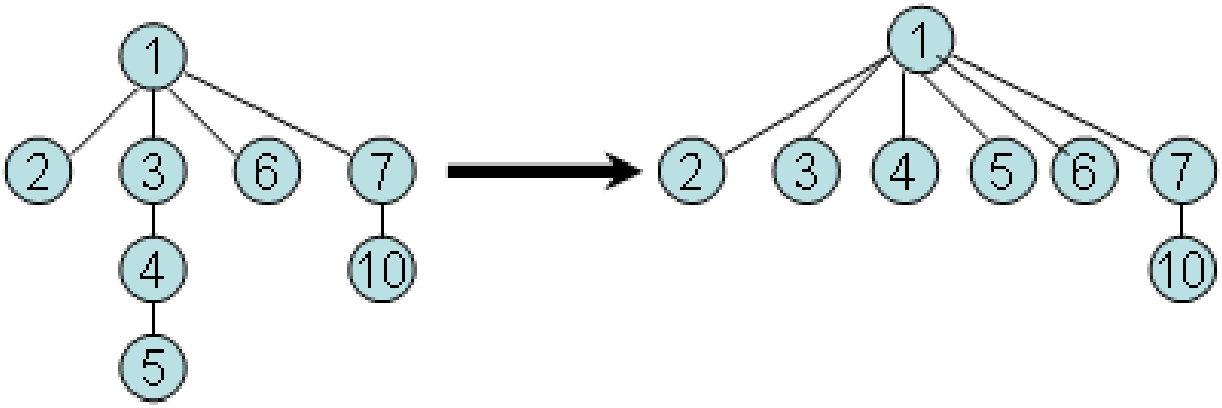
(有的人喜欢**fa[x]=0,**但很危险，需要最后查找一遍，否则有的根**fa[x]=0(**没有查找过**),**有的**fa[x]=x**（查找过）**;)**

三种操作：

Find(x)：查找x的树根

```
1. int find(int x){
2.     if(fa[x]==x) return x;
3.     return find(fa[x]);
4. }
```

查找find(i)的优化:路径压缩:



三种操作：

Find(x)：查找x的树根

```
1. int find(int x){
2.     if(fa[x]==x) return x;
3.     fa[x]=find(fa[x]);  //路径压缩
4.     return fa[x];
5. }
```

三种操作：

Find(x)：查找x的树根

```
1. int find(int x){
2.     if(fa[x]==x) return x;
3.     return fa[x]=find(fa[x]);//路径压缩
4. }
```

或者：

```
int find(int x){
     return fa[x]==x?x:fa[x]=find(fa[x]);
}
```

三种操作：

Union(a, b): 合并**x**和**y**所在的不同集合。

```
x=find(a);
y=find(b);
if(x!=y) p[x]=y;
```

合并的优化改进：

1、按树的结点个数合并
2、按树的高度合并

最后集合的个数就是有多少个结点的树根：
fa[x]=x
如果初始值是0
没有查找过的还是0
最好初始化为本身fa[x]=x

1. 读入边的顶点u，v
2. 查找u所在集合的树根结点x
3. 查找v所在集合的树根结点y
4. 如果x不等于y，合并x和y的集合
5. 最后统计有多少棵树即可

n个结点，m次操作：
并查集算法的时间复杂度O(4m)。

典型例题：

1346 【例4-7】亲戚(relation)
1385 团伙(group)
1347 【例4-8】格子游戏
1386 打击犯罪(black)