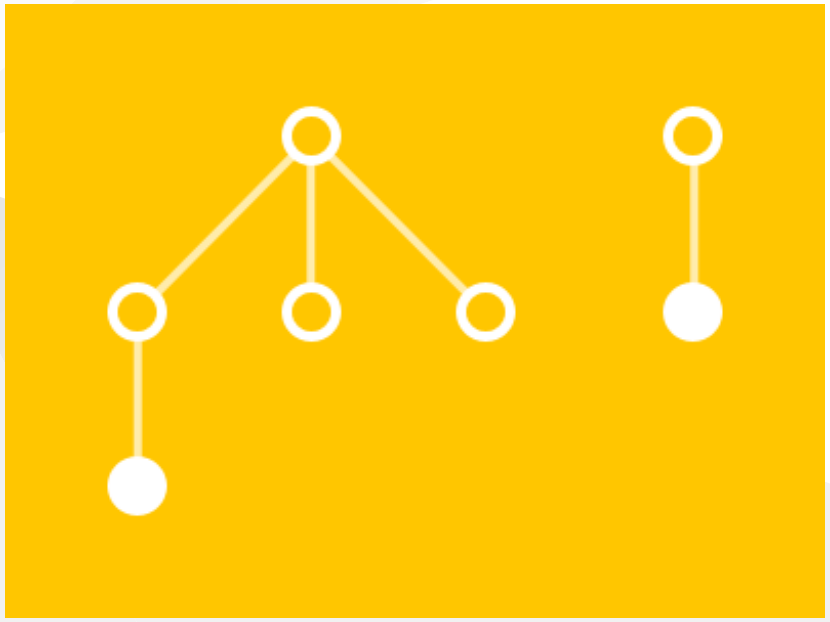


并查集

Union-Find Disjoint-set

19 音工 王睿

问题引出

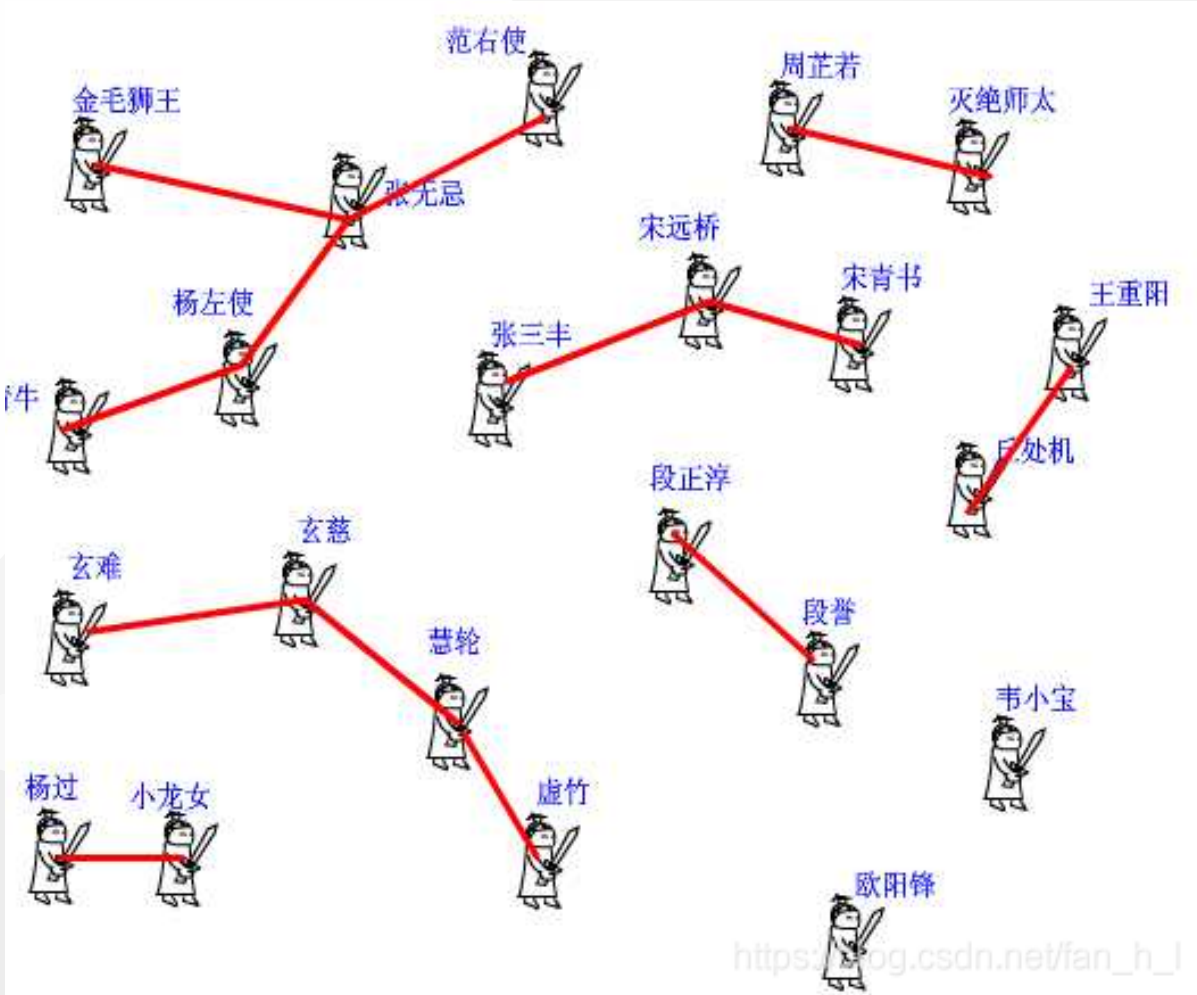
江湖上散落着各式各样的大侠，他们信奉**朋友的朋友就是我的朋友**，只要是能通过朋友关系串联起来的，不管拐了多少个弯，都认为是自己人。

两个原本互不相识的人，如何判断是否属于一个帮派（朋友圈）？

一个比较直接的思路是：

在各自的帮派中，选择一个帮主。

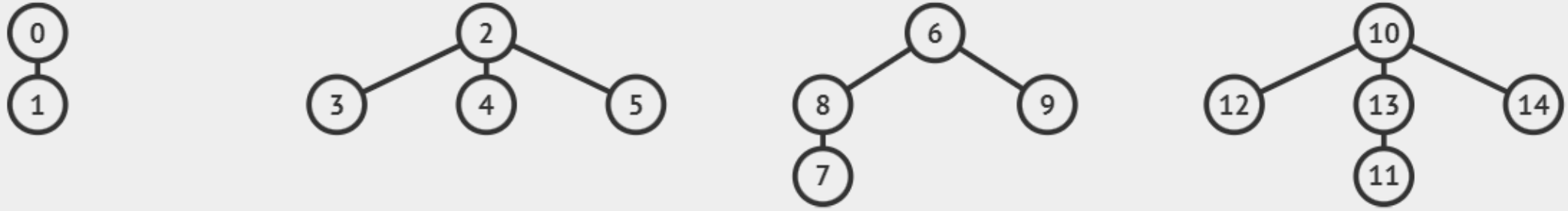
只要A和B的帮主是同一个人，那么A和B就是朋友。

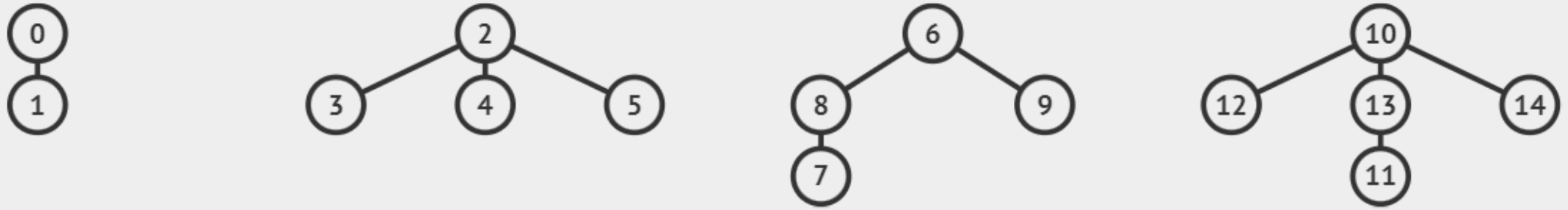


并查集概念

并查集是一种树型的数据结构，用于处理一些不相交集合的合并及查询问题。

并查集的思想是用一个数组表示了整片森林（parent），**树的根节点唯一标识了一个集合**（根节点作为**集合的代表元素**），我们只要找到了某个元素的的树根，就能确定它在哪个集合里。





用上图的数据结构表示前面提到的“帮派”：

- 一棵树为一个帮派，树的节点为帮派的成员
- 树的根为该帮派的帮主

操作

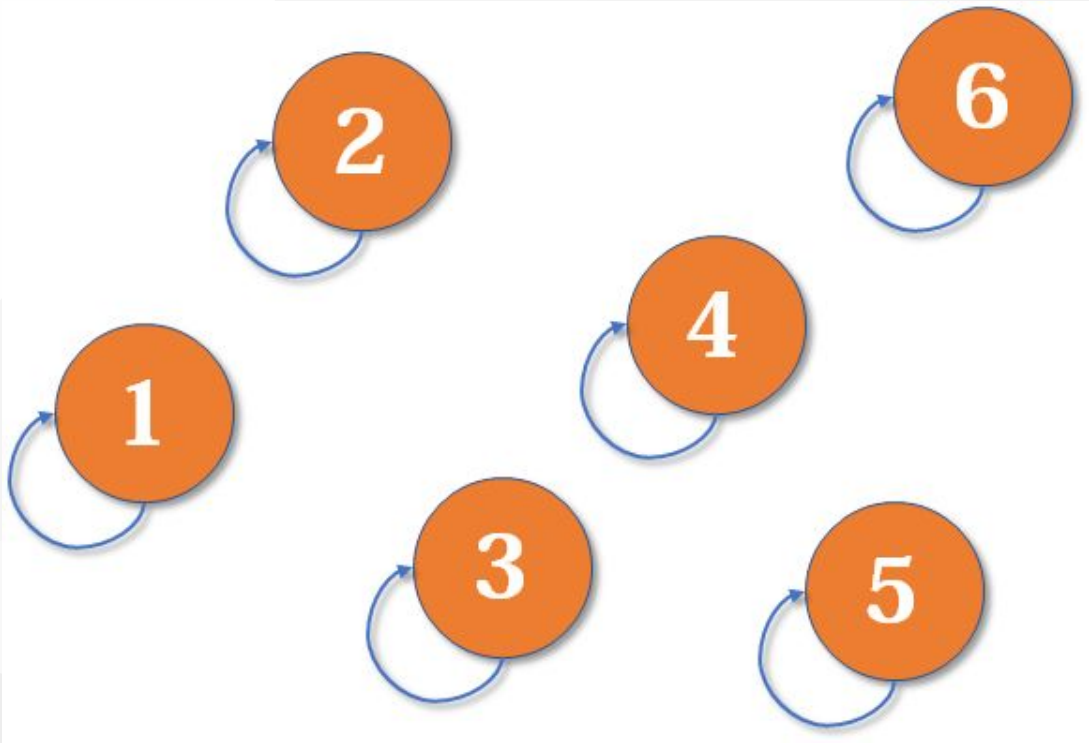
- 合并（Union）：把两个不相交的集合合并为一个集合。
- 查询（Find）：查询元素所在集合（的代表元素）。

如何运作

最开始，各元素各自为一个集合

互相之间都不认识。

各元素的父节点为自身，各元素所在集合的根节点为自身

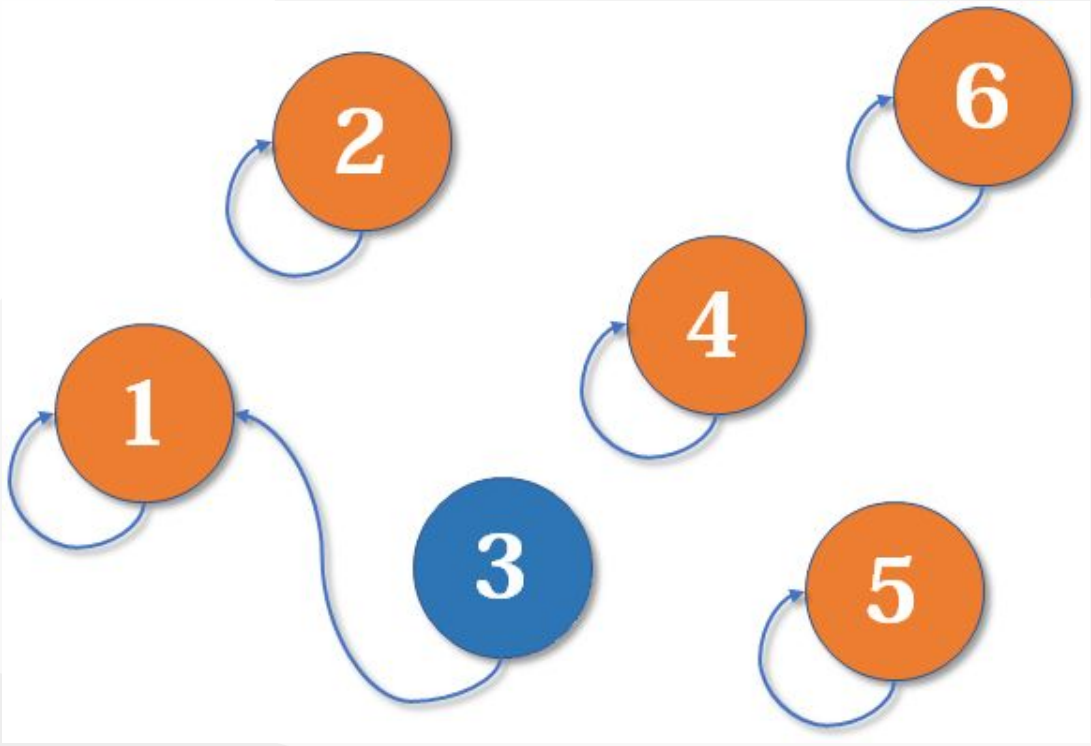


如何运作

现在得知1和3认识

那么3的父节点变为1（这里谁是父节点不重要）

该操作合并1和3所在的集合

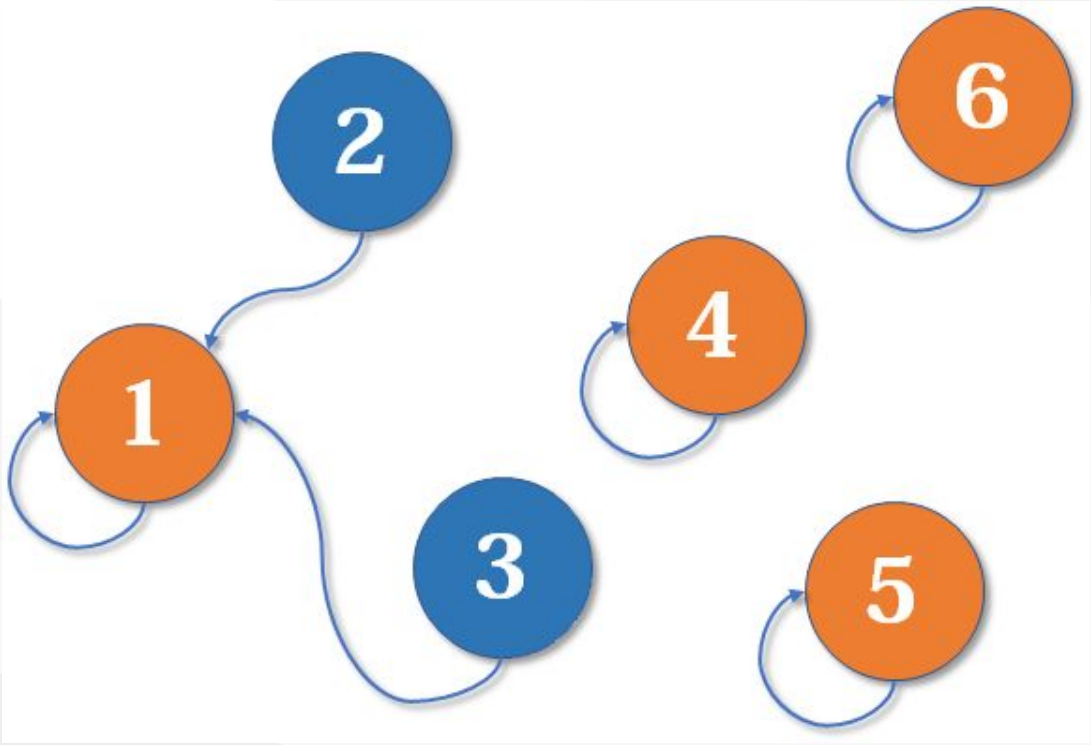


如何运作

现在得知3和2认识

在这里不直接把3作为2的父节点，而是将3的**根节点**1，作为2的父节点

该操作合并3和2所在的集合

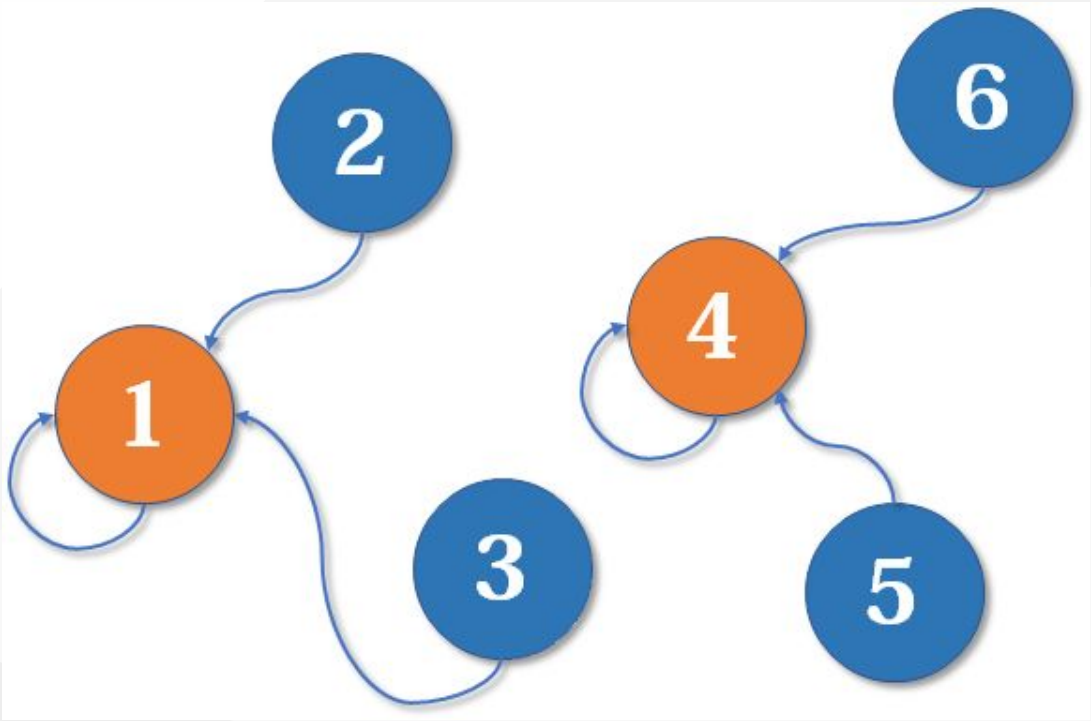


如何运作

现在我们假设4、5、6也进行了如上合并过程

目前我们得到了两颗树，1~6共分为了两个“朋友圈”

两颗树之间互相不认识，同一颗树内互相认识

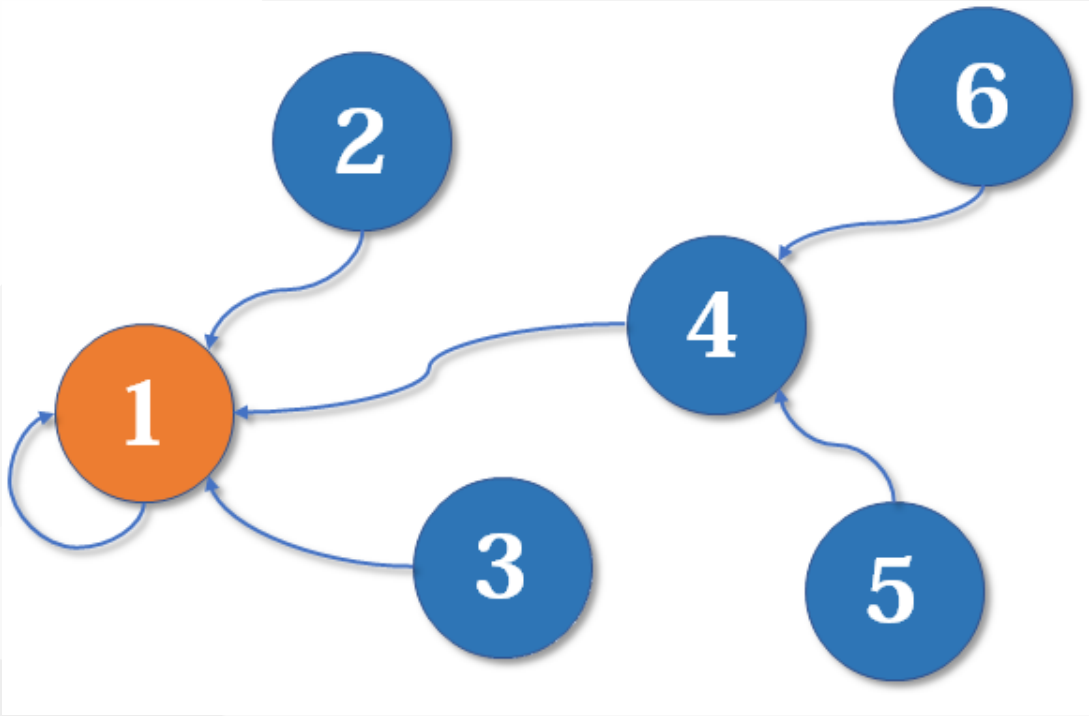


如何运作

现在得知2和6认识

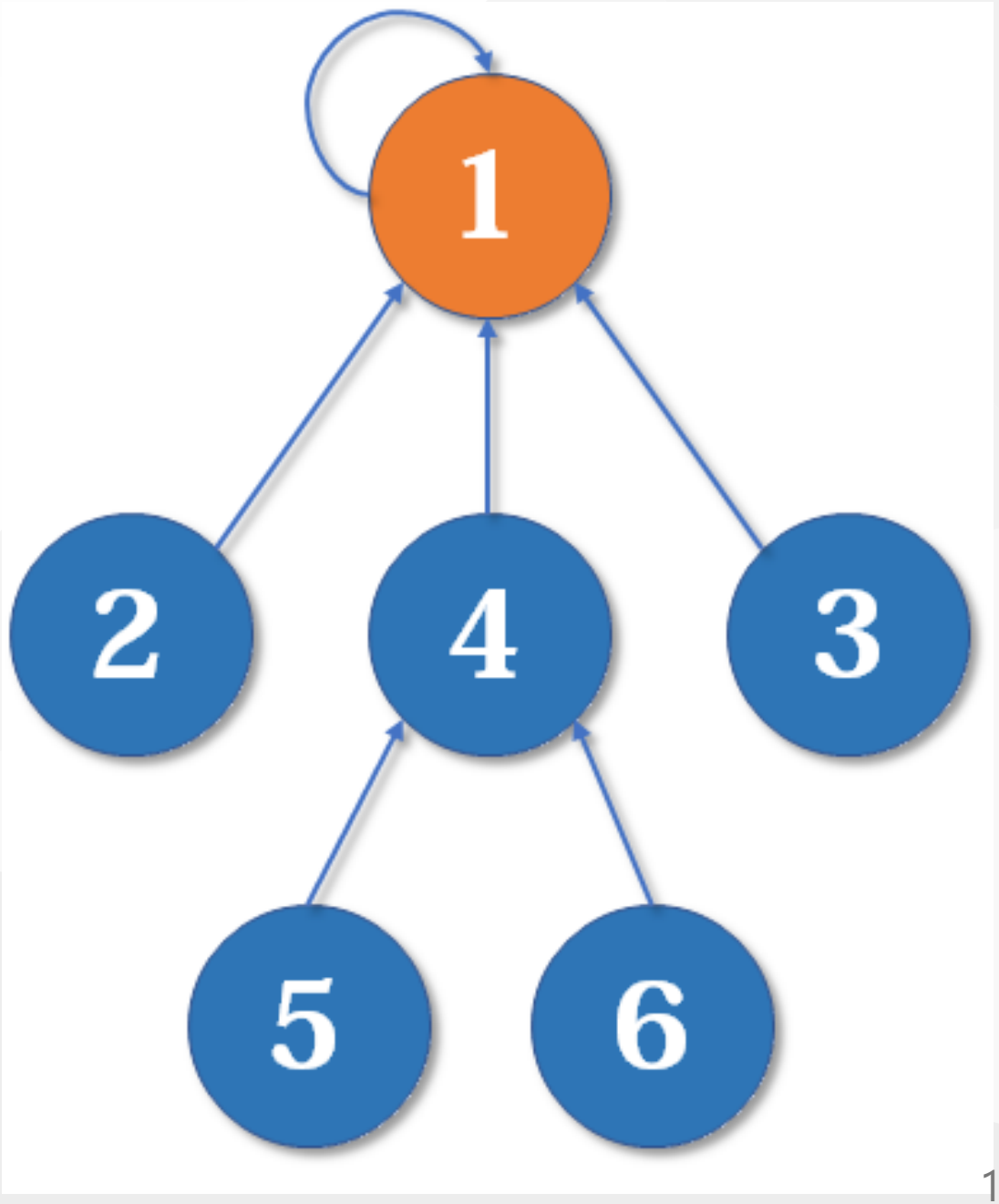
同上，用2的**根节点**（1），作为6的**根节点**（4）的父节点。

该操作合并2和6所在的集合



如何运作

这是一个树状的结构。要寻找集合的**代表元素**，只需要一层一层往上访问父节点（图中箭头所指的圆），直达树的根节点（图中橙色的圆）即可。根节点的父节点是它自己。我们可以直接把它画成一棵树：

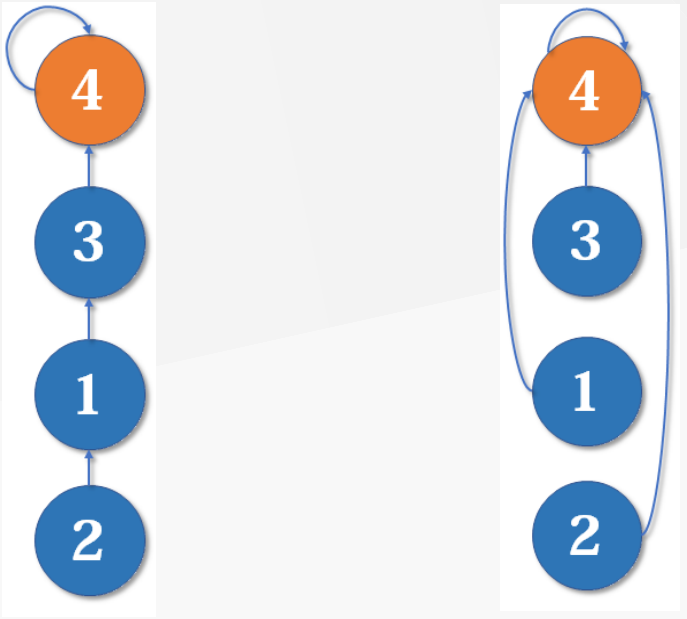


路径压缩

最简单的并查集效率是比较低的

如果存在一棵长链形的树，并且在合并的过程中，链的长度逐渐增加，那么找到根节点这一操作的开销就越来越大

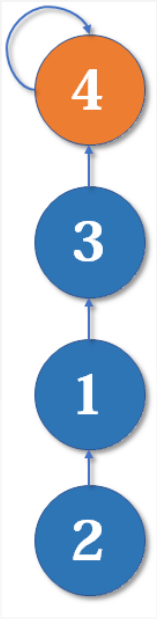
我们希望缩短左图的路径，使其变为右图，这样在每次寻找根节点时，就可以走过尽可能短的路径

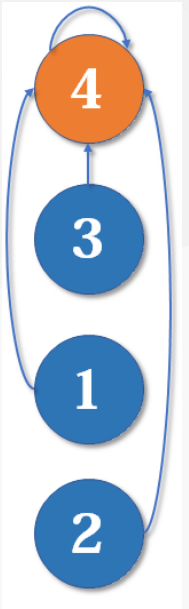


路径压缩

只要我们在查询的过程中，把沿途的每个节点的父节点都设为根节点即可

从2开始寻找根节点的过程中，依次将节点4、3、1、2的父节点都设置为4

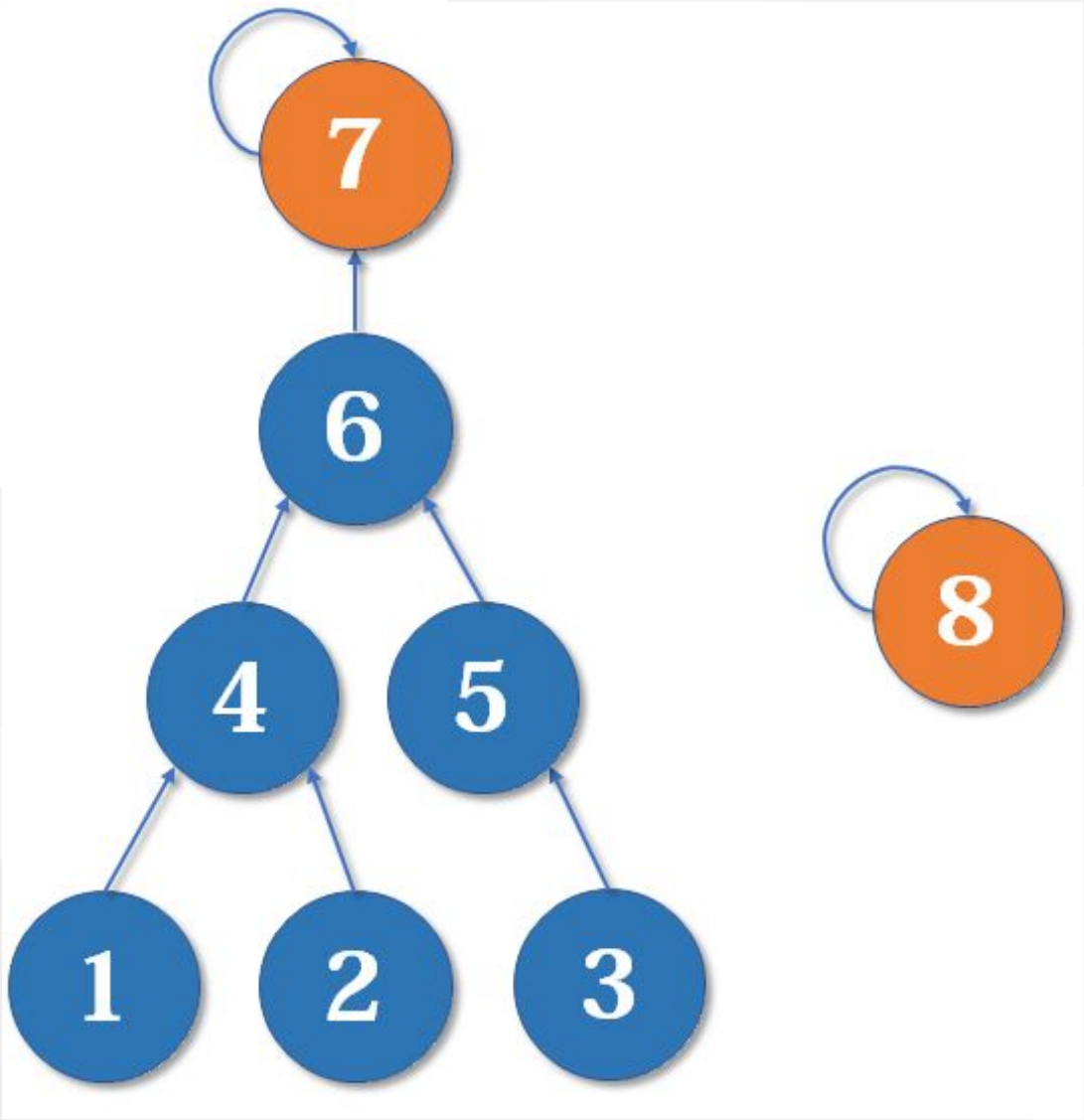




按秩合并

由于路径压缩只在查询时进行，也只压缩一条路径，所以并查集最终的结构仍然可能是比较复杂的。

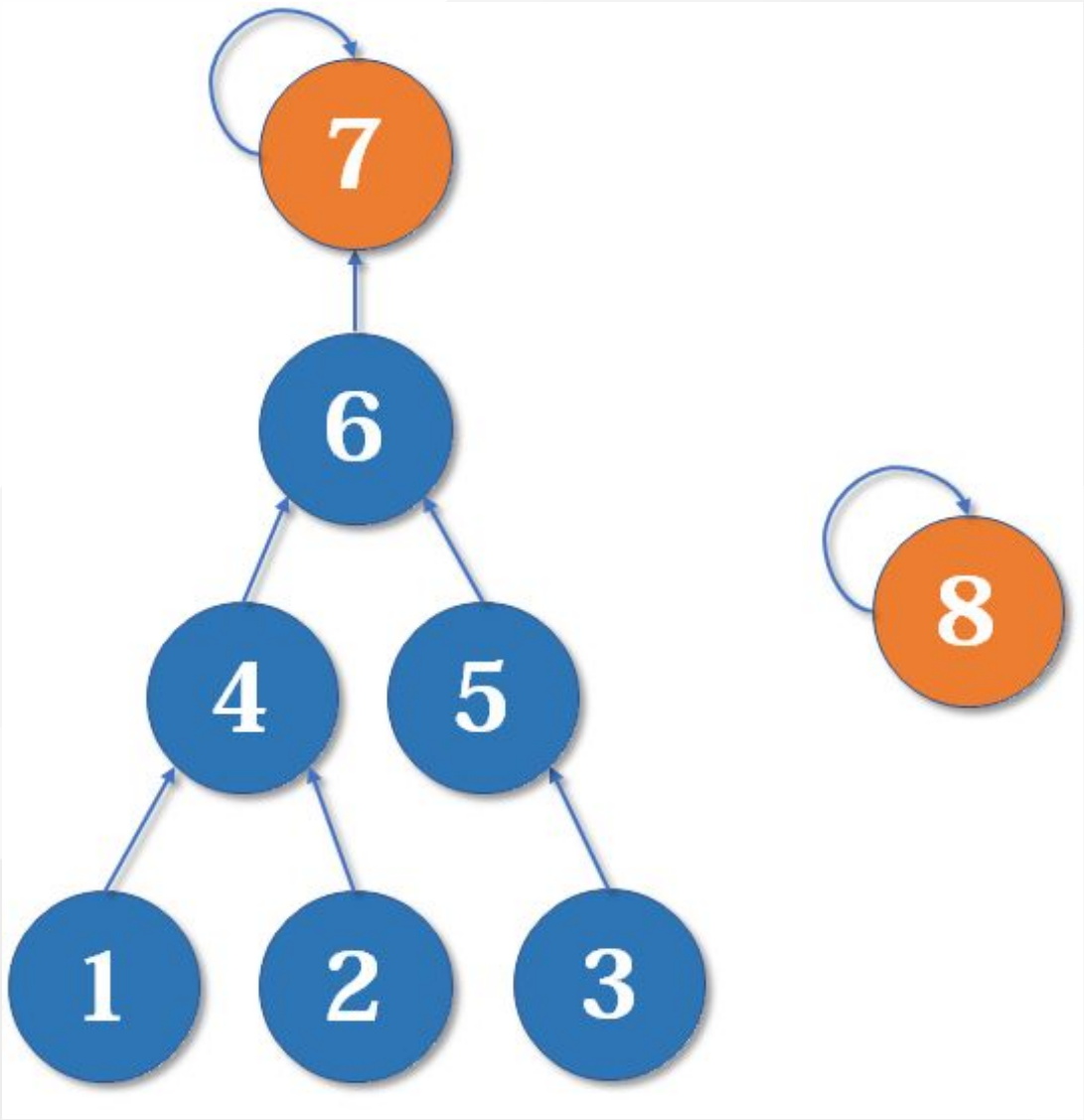
例如，现在我们有一棵较复杂的树需要与一个单元素的集合合并：



按秩合并

假如这时我们要合并7、8

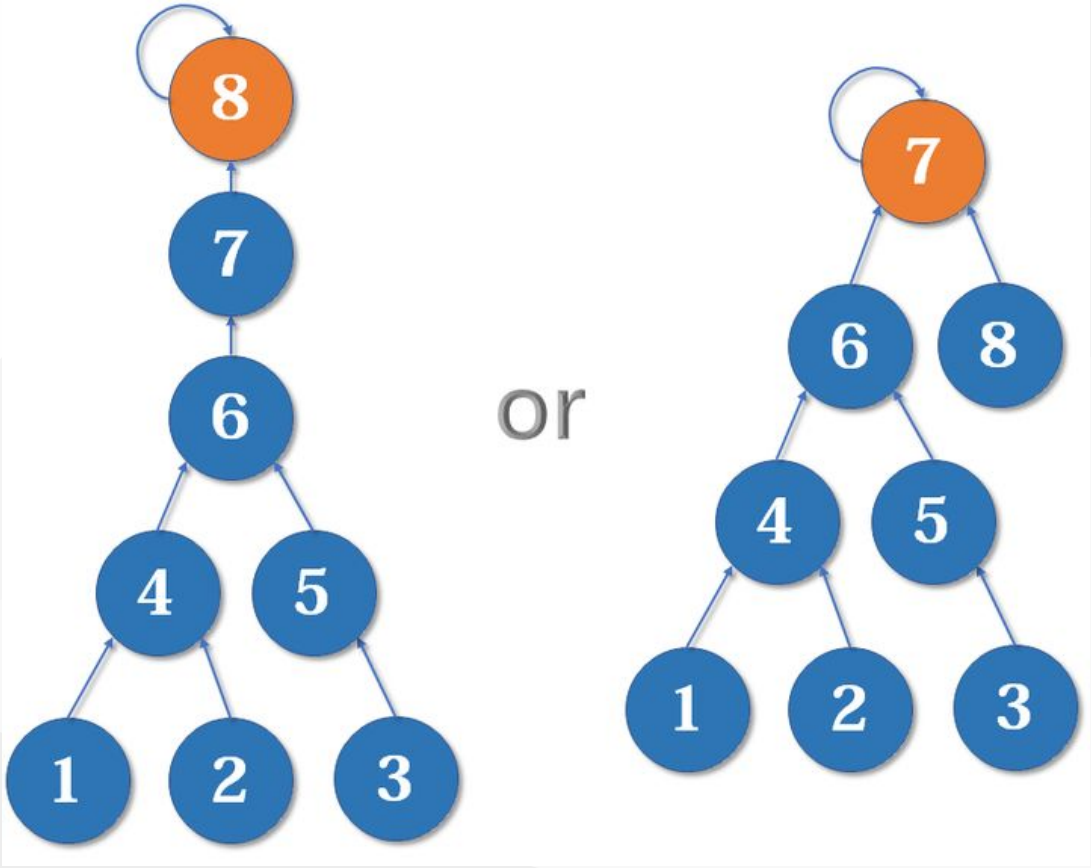
是把7的父节点设为8好，还是把8的父节点设为7好呢



按秩合并

如果把7的父节点设为8，会使树的深度加深，原来的树中每个元素到根节点的距离都变长了，之后我们寻找根节点的路径也就会相应变长。

而把8的父节点设为7，则不会有这个问题，因为它没有影响到不相关的节点。



按秩合并

- 什么是秩?
  有两个定义：树的深度（未路径压缩时）；集合的大小。
- 秩的存储?
  集合的秩一般存在代表元上，即树根上。
- 怎么实现?
  合并两个集合时，将秩小的树根作为秩较大的树根的子节点。
- 复杂度?
  单独使用按秩合并，Get的复杂度是O（logN）；
  同时使用路径压缩和按秩合并，Get的复杂度是O(α（N）)；
  α（N）是反阿克曼函数，是比logN增长还慢的函数。
- **一般情况只使用路径压缩就可以了。**

可视化

这里有一个数据结构和算法动态可视化网站

VISUALGO

感兴趣的同学可以动手操作试一试

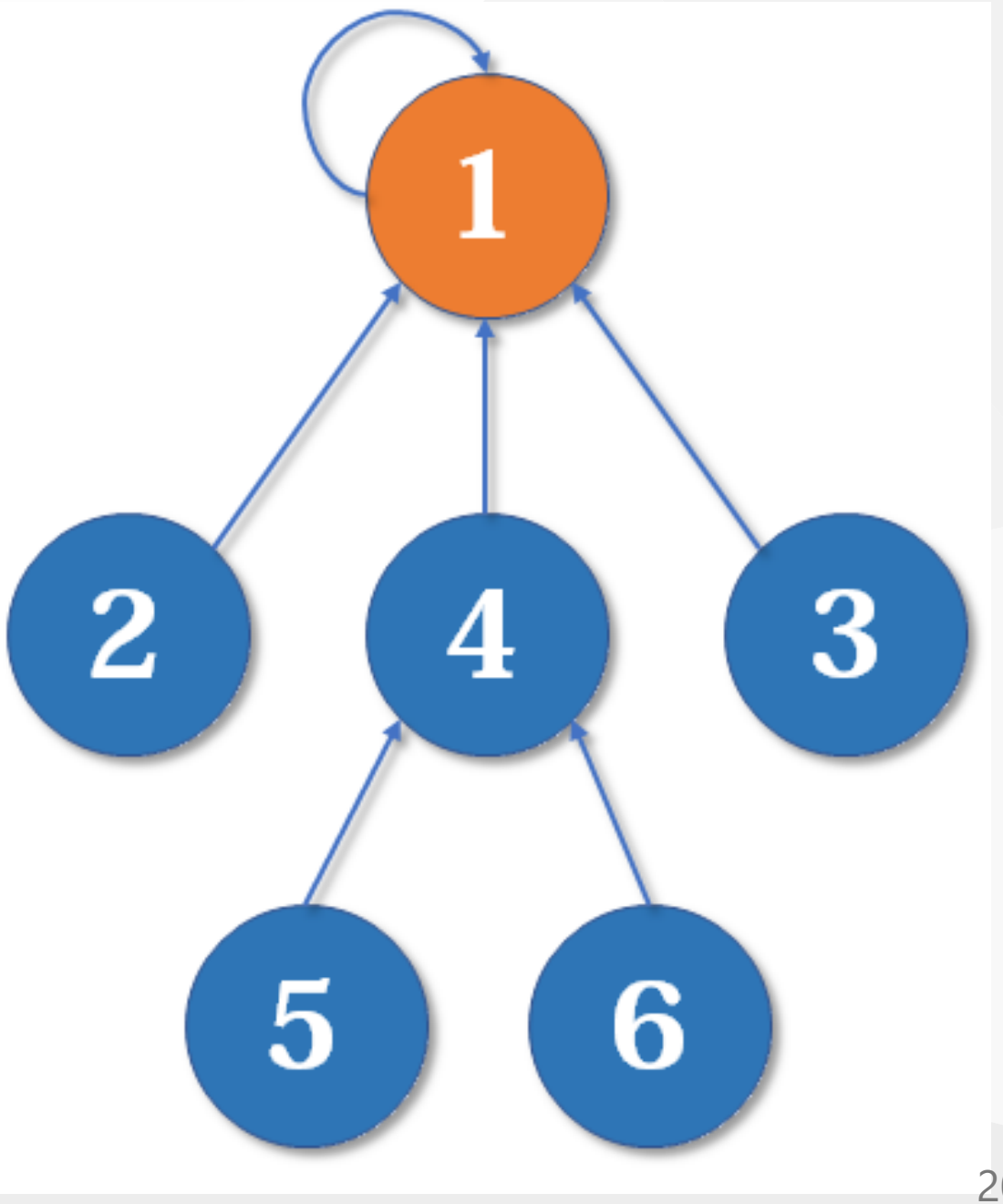
表示方式

构建上述的数据结构，只需存储每个节点的父节点即可

数组中，`fa[x]` 表示元素 `x` 的父节点，根节点的父节点为自己，即 `fa[root] = root`

右图所示的数据结构表示为

x	1	2	3	4	5	6
fa[x]	1	1	1	1	4	4



代码实现（路径压缩）

```
int fa[MAXN];
void init() {
    for (int i=1; i<=MAXN; i++) {
        fa[i]=i;
    }
}
int get(int x) {
    if (fa[x]==x) return x;
    return fa[x]=get(fa[x]); // 在查询根节点的过程中，进行路径压缩
}
void merge(int u,int v) {
    int p,q;
    p = get(u);
    q = get(v);
    if(p!=q) {
        fa[p]=q;
    }
}
```

例题

今天zjj要举办一个派对，但是zjj不知道需要多少张桌子来安排派对。zjj不想让不认识的人坐在一起，否则会影响友谊。

比如如果A认识B，B认识C，那么ABC三个人就可以坐在同一张桌子上；但如果A认识B，C认识D，那我们就默认AB和CD不认识，需要准备两张桌子。现在zjj需要你帮他计算出，我们一共需要多少张桌子。

输入

第一行是n和m，表示人数和关系数量
下面m行，每一行有一个u，v表示u和v之间相互认识

输出

一个整数，需要多少张桌子

例题——思路

标准的并查集。

先根据输入数据进行merge操作，得到节点间的关系（每个节点的父节点）

然后寻找父节点为本身的节点个数（根节点的个数）

根节点的个数即为集合的个数

例题——代码

```
init();
for (int i=1; i<=m; i++){
    merge(u,v); // 根据输入关系合并集合
}
cnt=0;
for (int i=1; i<=n; i++){
    if (fa[i]==i) { // 判断根节点的个数，即集合的个数
        cnt++;
    }
}
printf("%d\n",cnt);
```

并查集的扩展

- “边带权”并查集
- “扩展域”并查集

“边带权”并查集

例：有一个划分成N列的星际战场，各列依次编号为1-N，有N艘战舰。 编号为1-N.其中第i号战舰处于第i列。

$N \leq 30000, M \leq 5 * 10^5$

有M条指令，每条指令格式为以下两种之一:

1. M i j,表示让第i号战舰所在列的全部战舰保持原有顺序，接在第j号战舰所在列的尾部。
2. C i j，表示询问第i号战舰与第j号战舰当前是否处于同一列中，如果在同一列中，它们之间间隔了多少艘战舰。

思路

把每一列战舰看做一个集合，用并查集维护，初始N个战舰构成N个独立的集合

考虑路径压缩前：

- fa[x]表示：排在x战舰前面的那个战舰的编号；
- 一个集合的代表：排在最前面的那个元素；
- 每条边上带权值1：两点间距离减一就是两点间间隔的战舰数量。

考虑路径压缩后：

- d[x]表示：x与f[x]之间的权值；
- 路径压缩时：d[x]更新为从x到树根的路径上的所有边权之和

“扩展域”并查集

例：动物王国中有三类动物A,B,C，这三类动物的食物链构成了有趣的环形。A吃B，B吃C，C吃A。现有N个动物，以1~N编号。每个动物都是A,B,C中的一种。用两种说法对这N个动物所构成的食物链关系进行描述：

- "1 X Y"，表示X和Y是同类。
- "2 X Y"，表示X吃Y。

对N个动物，用上述两种说法，说出K句话，这K句话真假不确定。一句话满足下列三条之一时，这句话就是假话，否则就是真话。

1. 当前的话与前面的某些真的话冲突，就是假话；
2. 当前的话中X或Y比N大，就是假话；
3. 当前的话表示X吃X，就是假话。

你的任务是根据给定的N（1 <= N <= 50,000）和K句话（0 <= K <= 100,000），输出假话的总数。

思路

把每个动物x拆成3个结点：同类域x_self，捕食域x_eat，天敌域x_enemy

每句真话做如下操作：

1. “x和y是同类”：合并x_self 与 y_self; x_eat 与 y_eat; x_enemy 与 y_enemy
2. “x吃y”：合并 x_self 与 y_enemy ; x_eat 与 y_self ; x_enemy 与 y_eat;

处理每句话前检查：

1. 与“x和y同类”矛盾：
   x_slef 与 y_enemy 在同一集合（x吃y）
   y_self 与 x_enemy 在同一集合（y吃x）
2. 与“x吃y”矛盾：
   x_self 与 y_self 在同一集合（x与y是同类）
   y_self 与 x_enemy 在同一集合（y吃x）

参考资料

19计科赵润泽的PPT

https://blog.csdn.net/fan_h_l/article/details/107241265

https://www.runoob.com/data-structures/union-find-basic.html

https://zhuanlan.zhihu.com/p/93647900/

https://blog.csdn.net/weixin_55355427/article/details/117364723

https://blog.csdn.net/qq_39759315/article/details/80859210

AK愉快~