# 并查集

## 解决的问题：

有N个元素编号0到n-1(或者1到n)，他们起初各自属于不同集合(或者说不同分类)，有两种操作：1任意2个元素合并到同一集合(把分类变为相同)；2询问某2个元素是否属于同一集合(类别是否相同)。

由于合并和询问次数有很多，显然合并时候，不能暴力的储存每个人和他一类的都有谁，然后询问的时候在逐个试。

正确解法是使用并查集这一数据结构，并查集实际是一棵森林，初始有n个不相连的离散的点。实现方式用数组比较简单，建立数组f[i]代表i的父亲节点是谁，f[i]==[i]代表没有父亲; 建立数组rank，rank[i]代表节点i的高度,初始是1。

操作1： 合并x和y,那就把x的父亲指向y,p[x]=y，同时修改高度rank[x]=rank[y]+1

但是这样存在问题是，在最坏情况下，树会变成一条长链，很不利于查询操作，树越高，我们越不希望看到。所以有个办法是每次合并x和y，先查询他们谁离自己的根节点近,比如x距离根近我们称它的树是：离根近的数，**把距离根近的树直接连接在距离根远的树的根上，就能有效减少并查集高度。**随着节点数增长，高度增长缓慢，这种方法称之为路径压缩或者启发式合并。

操作2：直接沿着x和y各自父节点走，看父节点是不是同一个即可。

并查集用路径压缩的方法，查询的平均时间复杂度是：O(α(n)) 它是[阿克曼函数](https://zh.wikipedia.org/wiki/%E9%98%BF%E5%85%8B%E6%9B%BC%E5%87%BD%E6%95%B0)反函数，它比log(n)还好，接近o(1).