# 第三讲：并查集

作者：马馨萍

## 基础并查集

### 简介

**并查集**是一种树型的**数据结构**，主要用于解决一些**元素分组**的问题。它管理一系列**不相交的集合**，并支持两种操作。

**合并**（Union）：把两个不相交的集合合并为一个集合。

**查询**（Find）：查询两个元素是否在同一个集合中。

#### 情景引入

帮派问题

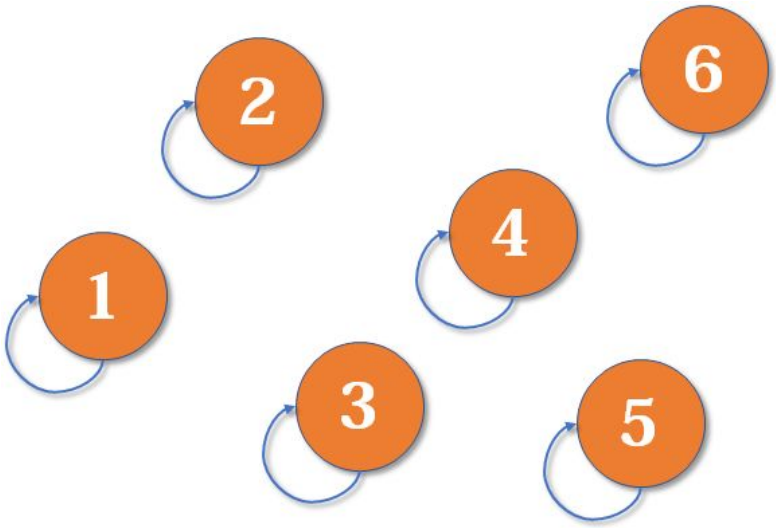
具体来说就是江湖上有各种各样的帮派，同一个帮派里的都是朋友，不同帮派的不是一路人，就免不了要打一架。

- 如何体现朋友的朋友就是朋友？
- 如果给出两个人，如何判断他们是不是同一个帮派？

  并查集的**重要思想**：将一个集合抽象为帮派，其中选一个**代表元素**，即为帮主。

```
int f[maxn];    //维护每个点的所属帮派的帮主
```

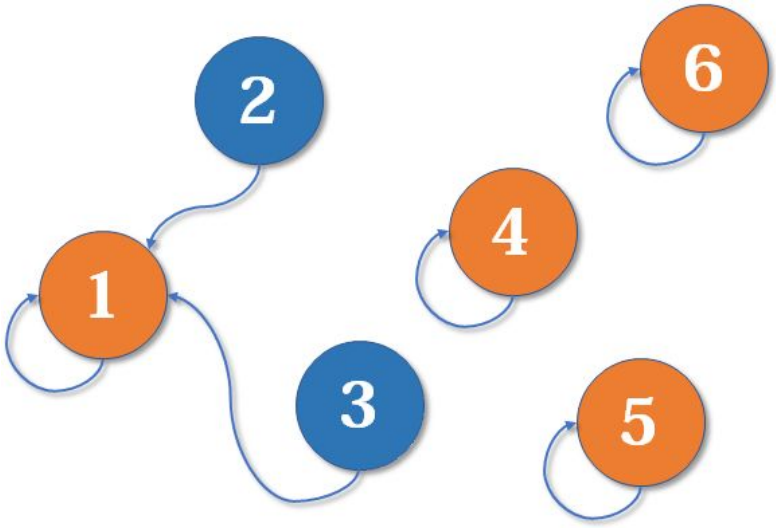
初始时，都是一个人行走江湖，每个人都自成一派。



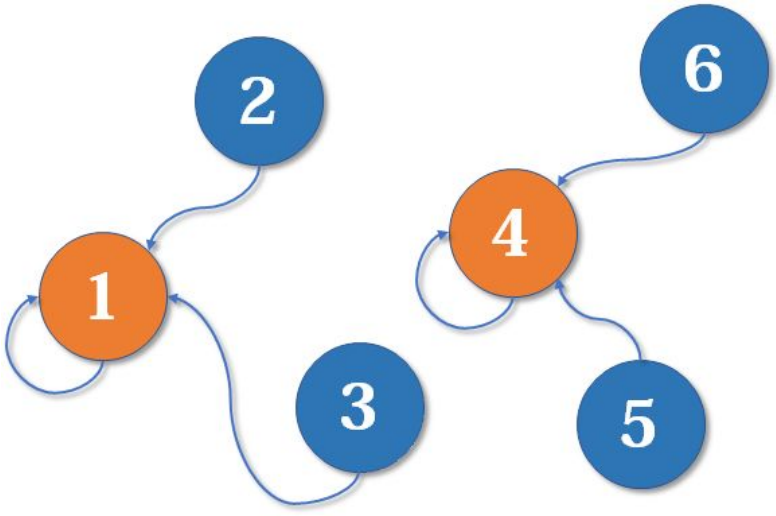

```
for(int i=1;i<=n;i++){
    f[i]=i;
}
```

接下来给出一些友好关系，会造成不同的帮派合并，拉帮结派其实就是对应的并查集的合并操作。注意每次的合并，都是一个帮派的代表元素（即帮主）出面的。

比方说此时1 2要结拜，1 3要结拜，然后2 3都认了1当大哥。

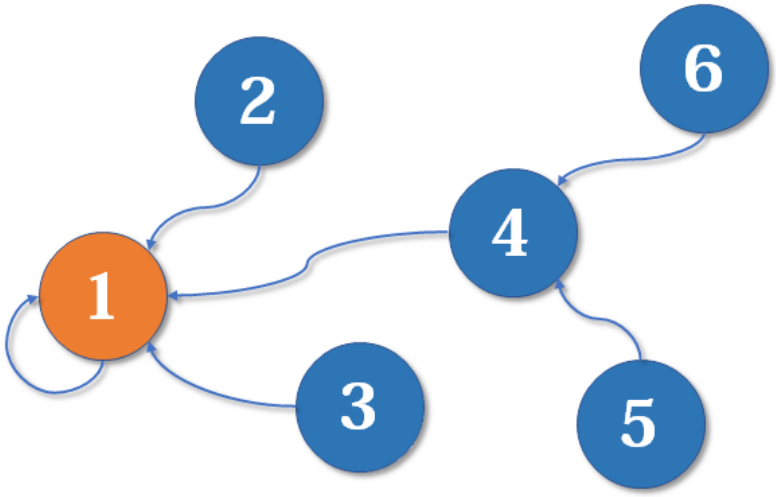


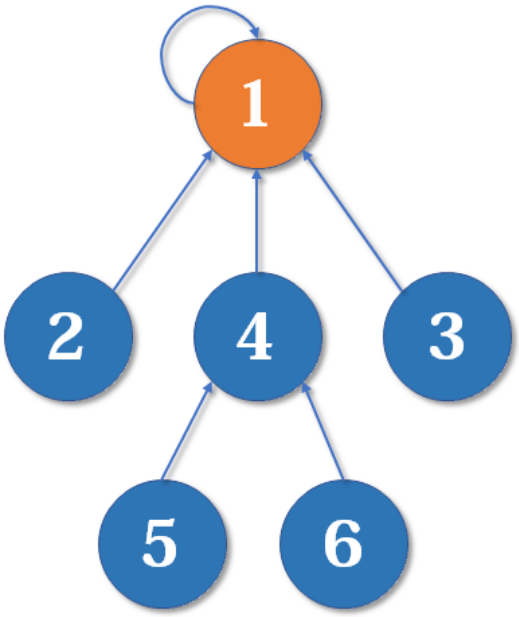

然后4 5 6合并，假定让4做帮主




接下来，如果要合并2和5所在的集合怎么办?

2找到它所在帮派的帮主--1，5找到它所在帮派的帮主--4，在1 4中选一个当帮主，另一个帮主去当小弟。

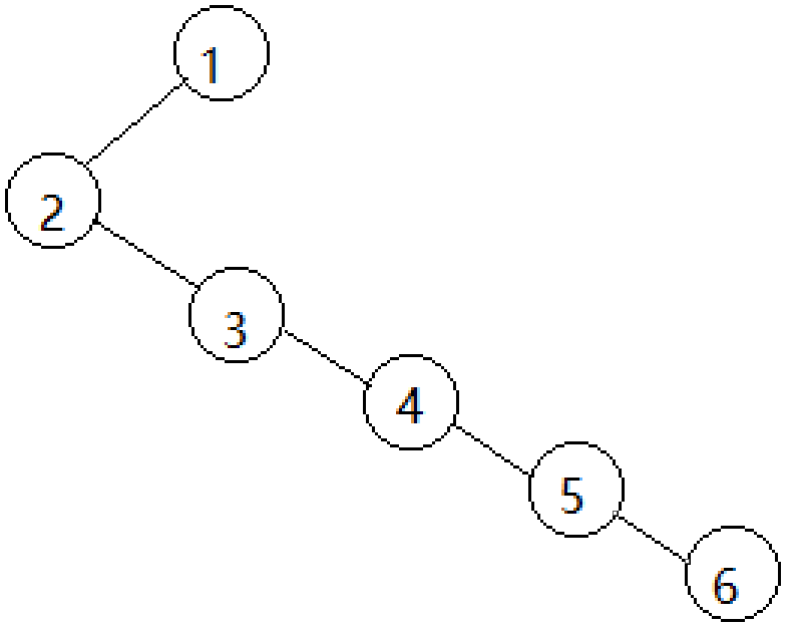


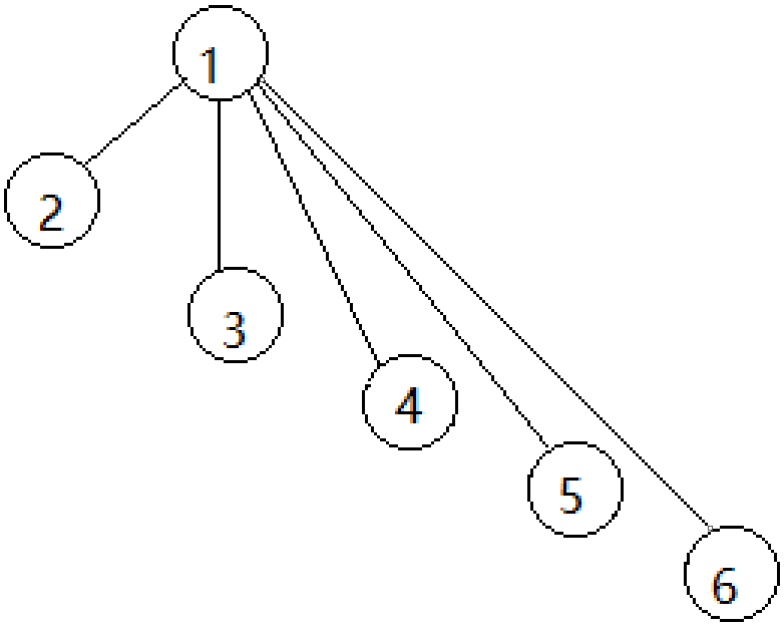




**查询**

```
int find(int x){    //找掌门函数
    if(f[x]==x) return x;  //如果自己就是根就返回自己
    return find(f[x]);     //否则就返回自己掌门的掌门
}
```

**路径压缩**



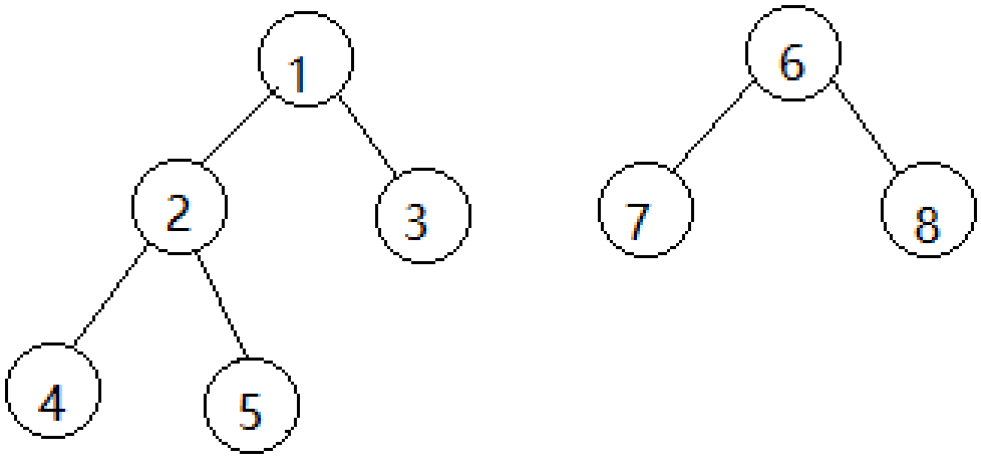

```
int find(int x){
    if(f[x]==x) return x;
    //下面这两句可以直接整合成：return f[x]=find(f[x]);
    f[x]=find(f[x]);
    return f[x];
}
```

**合并**

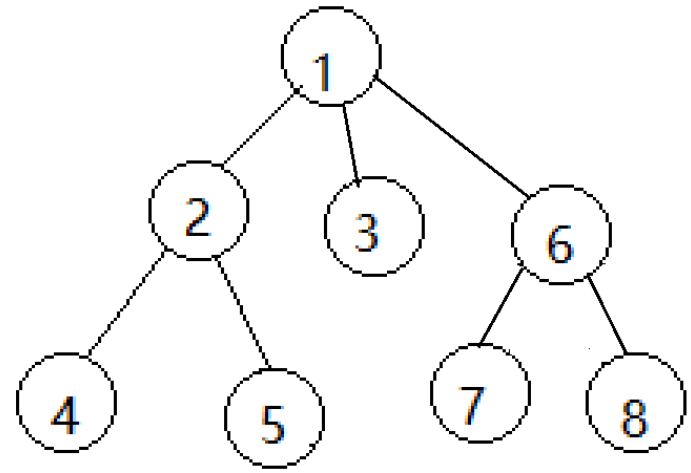
```
void merge(int x,int y){
    int fx=find(x);   //找到x的根
    int fy=find(y);   //找到y的根
    if(fx!=fy) f[fy]=fx;  //如果根不同，说明x、y不在同一个集合，就合并。这里让fy的集合做为fx的子树
}
```

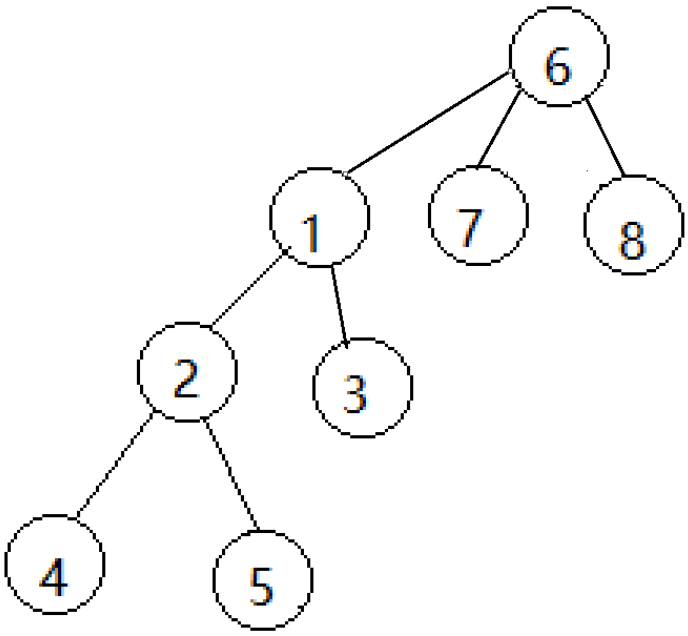
**按秩合并**

秩--树的高度




两种合并方式：




```
int my_rank[maxn];   //C++ STL的std::rank模板
void my_union(int x,int y){   //union是关键字
    x=find(x);   //后面用不到这个x,y，我们让他们表示他们的最大的老大，比较秩
    y=find(y);
    if(x==y) return;
```

```
    if(my_rank[x]<my_rank[y]){  //秩不变
        f[x]=y;
    }
    if(my_rank[x]==my_rank[y]){
        f[x]=y;
        my_rank[y]++;
        //或者是f[y]==x;rank[x]++;
    }
    if(my_rank[x]>my_rank[y]){
        f[y]=x;
    }
}
```

**复杂度**

- 如果只有路径压缩，或者只有按秩合并，并查集单次操作的复杂度都是 $O(logn)$ 。
- 但是如果两个同时使用，复杂度可以看做是常数。

## 例题选讲

例一：P1551 亲戚

若某个家族人员过于庞大，要判断两个是否是亲戚，确实还很不容易，现在给出某个亲戚关系图，求任意给出的两个人是否具有亲戚关系。

规定：x和y是亲戚，y和z是亲戚，那么x和z也是亲戚。如果x,y是亲戚，那么x的亲戚都是y的亲戚，y的亲戚也都是x的亲戚。

```
#include<bits/stdc++.h>
using namespace std;
#define maxn 5005

int n,m,p;
int f[maxn];

int find(int x){
    if(f[x]==x) return x;
    return f[x]=find(f[x]);
}

void merge(int x,int y){
    int fx=find(x);
    int fy=find(y);
    if(fx!=fy) f[fy]=fx;
}

int main(){
    cin>>n>>m>>p;
    for(int i=1;i<=n;i++) f[i]=i;
    int x,y;
    for(int i=1;i<=m;i++){
        cin>>x>>y;
        merge(x,y);
    }
    for(int i=1;i<=p;i++){
        cin>>x>>y;
```

```
        if(find(x)==find(y)) cout<<"Yes"<<endl;
        else cout<<"No"<<endl;
    }
    return 0;
}
```

例二：P1111 修复公路 ——并查集判断连通性

给出A地区的村庄数$N$，和公路数$M$，公路是双向的。并告诉你每条公路的连着哪两个村庄，并告诉你什么时候能修完这条公路。问最早什么时候任意两个村庄能够通车，即任意两个村庄都存在一条或多条路将其联通。

## 带权并查集

例三：P1196 [NOI2002] 银河英雄传说

战舰也依次编号为$1,2,\ldots,30000$。

合并指令：`M i j`，含义为第 $i$ 号战舰所在的整个战舰队列，作为一个整体（头在前尾在后）接至第 $j$ 号战舰所在的战舰队列的尾部。

询问指令：`C i j`。该指令意思是，询问电脑，第 $i$ 号战舰与第 $j$号战舰当前是否在同一列中，如果在同一列中，那么它们之间布置有多少战舰。如果第 $i$号战舰与第 $j$ 号战舰当前不在同一列上，则输出 -1。

```
//银河英雄传说
#include<bits/stdc++.h>
using namespace std;
#define maxn 30005

const int n=30000;
int T;
int f[maxn];
int num[maxn],dis[maxn];

int find(int x){
    if(x==f[x]) return x;
    else{
        int fa=f[x];
        find(fa);
        dis[x]+=dis[fa];
        f[x]=f[fa];
        return f[x];
    }
}

void merge(int x,int y){
    int fx=find(x),fy=find(y);
    f[fx]=fy;
    dis[fx]=num[fy];
    num[fy]+=num[fx];
}

int main(){
    cin>>T;
    int i;
```

```
    for(i=1;i<=n;i++){
        f[i]=i;
        dis[i]=0;
        num[i]=1;
    }
    int x,y;
    char flag;
    for(i=1;i<=T;i++){
        cin>>flag>>x>>y;
        if(flag=='M'){
            merge(x,y);
        }
        if(flag=='C'){
            int fx=find(x),fy=find(y);
            if(fx!=fy) cout<<"-1"<<endl;
            else cout<<abs(dis[x]-dis[y])-1<<endl;
        }
    }
    return 0;
}
```

### 种类并查集

例四：P2024 [NOI2001] 食物链

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。有两种说法：

- 第一种说法是 `1 X Y`，表示 X 和 Y 是同类。
- 第二种说法是 `2 X Y`，表示 X 吃 Y 。

说出 K 句话，这 K 句话有的是真的，有的是假的。

- 当前的话与前面的某些真的话冲突，就是假话
- 当前的话中 X 或 Y 比 N 大，就是假话
- 当前的话表示 X 吃 X，就是假话

求假话的总数。

```
//食物链
#include<bits/stdc++.h>
using namespace std;
#define maxn 50005

int n,k;
//i->A  i+n->B  i+n+n->C
int f[maxn*3];

int find(int x){
    if(x==f[x]) return x;
    return f[x]=find(f[x]);
}

void merge(int x,int y){
    int fx=find(x),fy=find(y);
    if(fx!=fy){
```

```
        f[fy]=fx;
    }
}

int main(){
    cin>>n>>k;
    int i;
    for(i=1;i<3*n;i++) f[i]=i;
    int flag,x,y;
    int cnt=0;
    for(i=1;i<=k;i++){
        cin>>flag>>x>>y;
        if(x>n||y>n){
            cnt++;
            continue;
        }
        if(flag==1){
            if(find(x)==find(y+n)||find(x)==find(y+n+n)){
                cnt++;continue;
            }
            merge(x,y);
            merge(x+n,y+n);
            merge(x+n+n,y+n+n);
        }
        else{
            if(find(x)==find(y)||find(x)==find(y+2*n)){
                cnt++;continue;
            }
            merge(x,y+n);
            merge(x+n,y+2*n);
            merge(x+2*n,y);
        }
    }
    cout<<cnt;
    return 0;
}
```