~~我认为~~最优雅的数据结构

——并查集

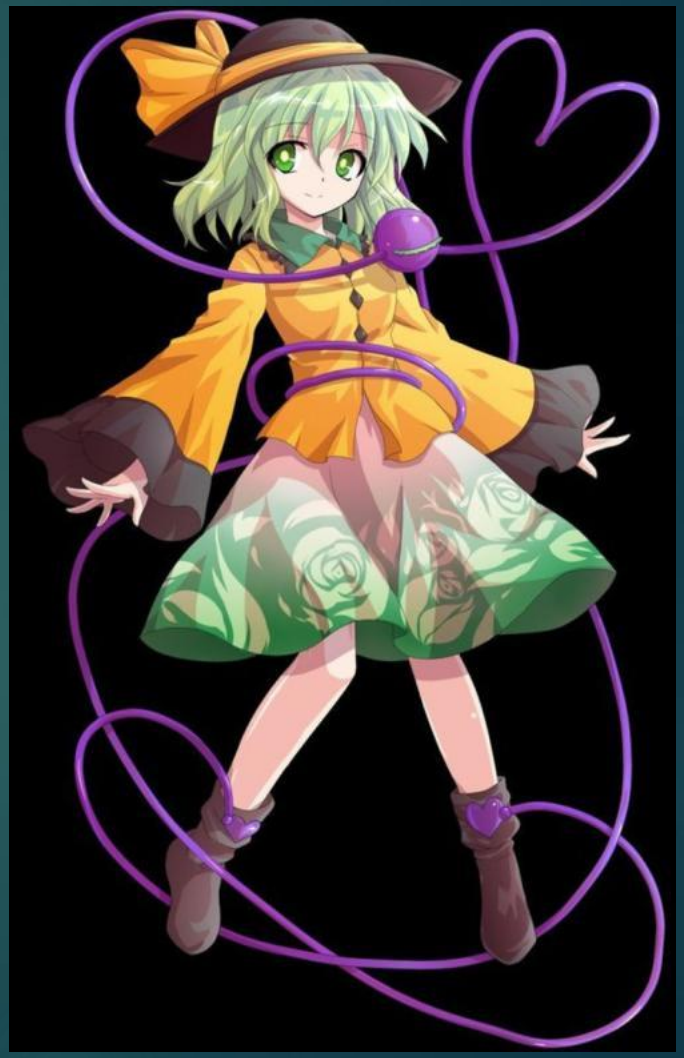
引言

- 规定：x 和 y 是朋友，y 和 z 是朋友，那么 x 和 z 也是朋友。如果 x, y 是朋友，那么 x 的朋友都是 y 的朋友，y 的朋友也都是 x 的朋友。

- 问题：现在给出若干个朋友关系，多次询问，求任意给出的两个人是否具有朋友关系。

第一轮抽象

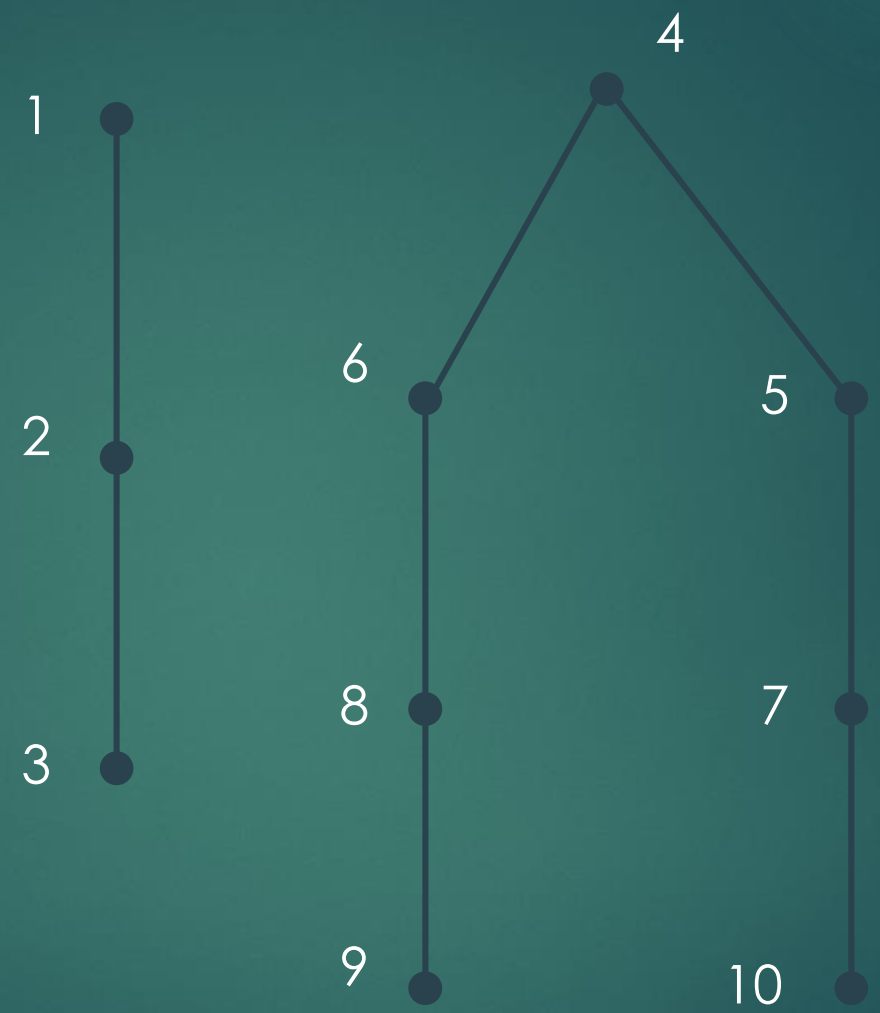
- 把每个人抽象成一个点，然后标号
- 当两个人是朋友，把这两个点用一条线连起来
- 问题转化成，给你一个图，询问两个点是否互相连通



一个栗子

- 关系：

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



缺陷

- 给计算机一个图，不管这个图结构如何，通过一个点判断这个点能到达哪些点，复杂度都是O(n)的。

- 也就是说如果给的人数和关系如果过多，同时需要多次询问，~~罚~~耗时会爆炸。

再次抽象

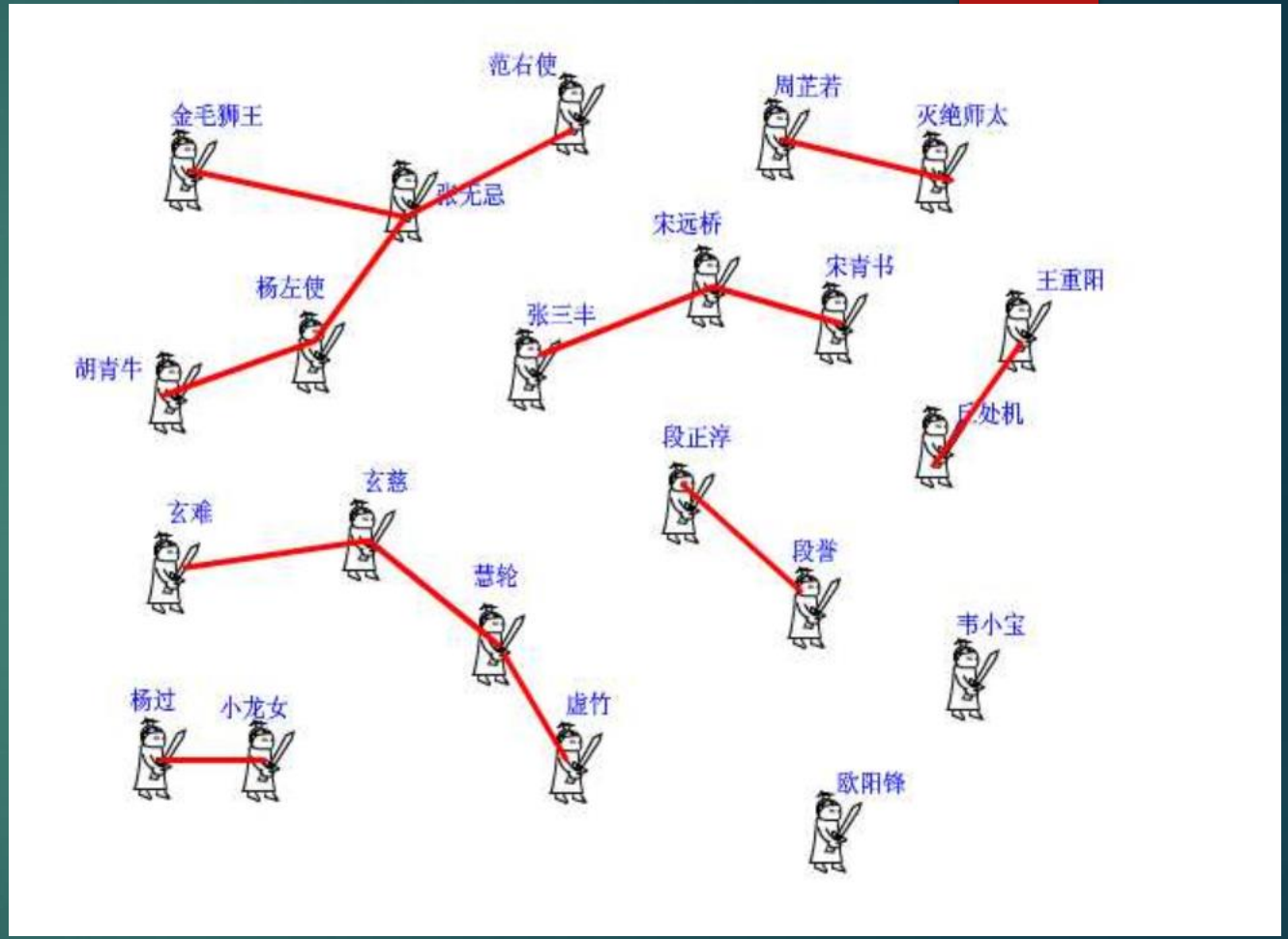
- 建图方法 GG了。
- 换一种方法？
- 考虑集合
- 初始时每个人各自当成一个集合
- 查询 x y 是否是朋友　　-> 查询 x y 是否属于同一集合
- x y 是朋友　　-> 将 x y 所属的集合合并

一些概念

- 并查集主要用来解决判断两个集合是否属同一个集合，以及把两个集合合并成一个集合的问题。
- 同属一个集合关系是一个等价关系，因为他满足等价关系的三个条件
- 其实我之前还看不懂这个的时候就学会并查集了（小声）

关于实现的栗子

- 江湖中有许多门派
- 每个人都有一个直系领导
- 领导也有他的领导
- 最终的领导叫掌门（即领导是他自己）
- 如何判断相同门派　-> 掌门相同



具体实现 —— 查

- 一个数组 pre[MAXN] -> pre[i] 表示 i 的上级
- 什么是掌门？　pre[i] == i;

```
void Find(int x)      ///查找x的掌门
{
    while(x != pre[x]){ ///如果上级不是掌门
        x = pre[x];     ///那就寻找上级的上级
    }

    return x;    ///掌门驾到！
}
```

具体实现 —— 并

- 看起来是一整个门派的掌门都变了
- 其实只要把其中一个门派的掌门变成另一个门派掌门的上级就好了

```
void Join(int x, int y) ///虚竹想和周芷若做朋友
{
    x = Find(x);      ///我老大是玄慈
    y = Find(y);      ///我老大是灭绝
    if(x != y){        ///两个人老大不一样
        pre[x] = y; ///灭绝当了玄慈的老大
    }
}
```

具体实现 —— 初始化

- 最开始每个人都是独行侠，各成门派

```
void ini()
{
    for(int i = 1; i <= maxn; i ++){
        pre[i] = i; ///自己做自己的老大
    }
}
```

小优化 —— 路径压缩

- 建立门派的过程是用 Join 函数两个人两个人地连接起来的，谁当谁的手下完全随机。最后的树状结构会变成什么样，我也无法预知，一字长蛇阵也有可能。这样查找的效率就会比较低

- 最理想的情况就是所有人的直接上级都是掌门，一共就两级结构，只要找一次就找到掌门了。哪怕不能完全做到，也最好尽量接近。这样就产生了路径压缩优化。

- 在查询掌门的过程中，把一路找到人的上级全都更新成掌门！

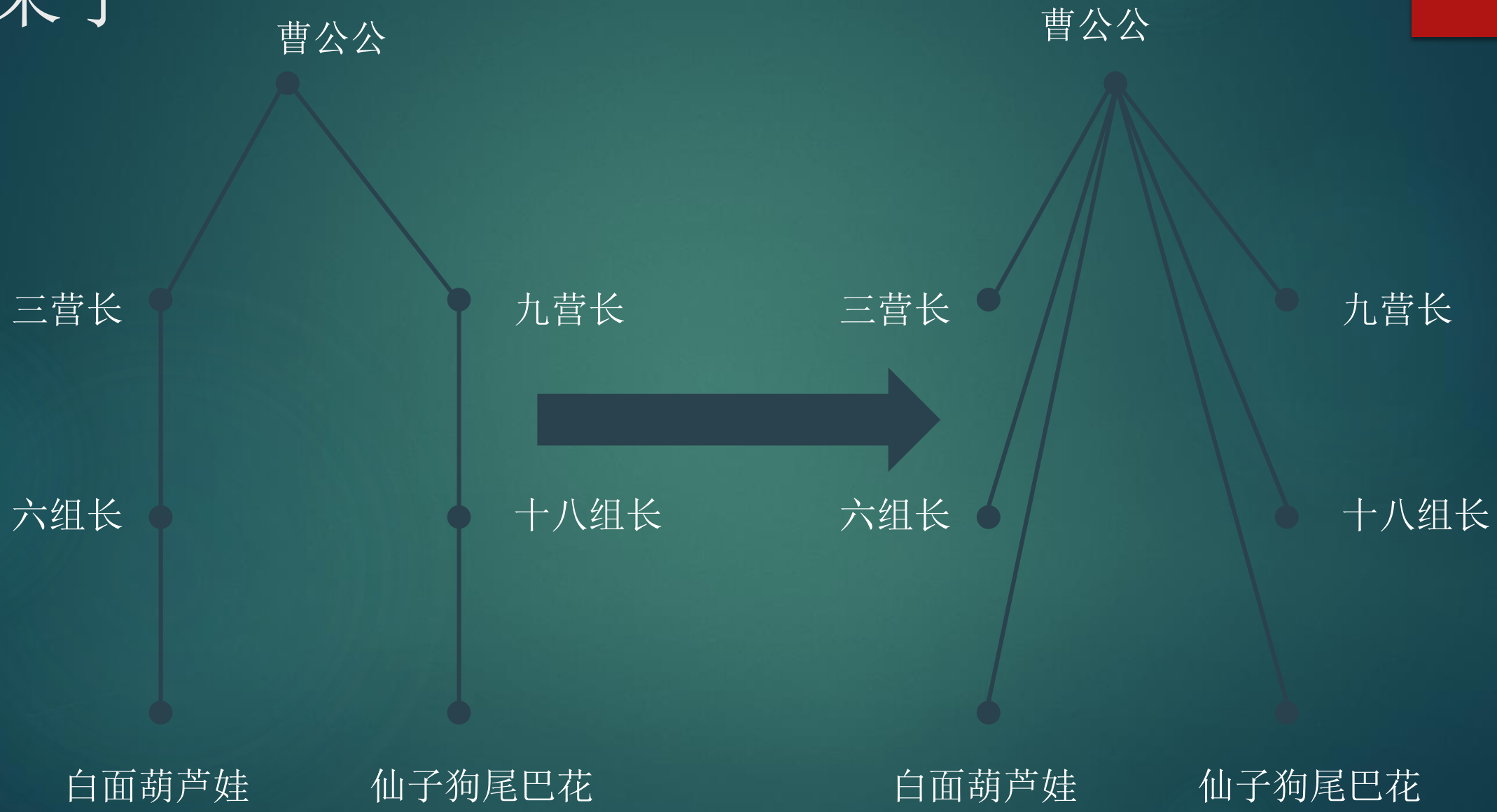
```
void Find(int x)
{
    int t = x;

    while(t != ne[t]){
        t = ne[t];
    }
    while(x != t){   ///多了个这个
        int q;

        q = ne[x];
        ne[x] = t;
        x = q;
    }

    return t;
}
```

栗子


曹公公
三营长
九营长
六组长
十八组长
白面葫芦娃
仙子狗尾巴花
曹公公
三营长
九营长
六组长
十八组长
白面葫芦娃
仙子狗尾巴花

完全体 —— 启发式合并

- 也就是合并的时候不是随机合并，而是把人数少的门派合并到人数多的门派之下。也有的按照深度来决定合并。

科普性质

- 完全体的并查集（路径压缩 + 启发式合并）的复杂度是阿克曼函数的反函数
- 大概就是说宇宙级别的数据我们只需要常数级别的时间就能解决！
- 这个复杂度是由**Tarjan男神**在1979年证明
- 这个证明我至今看不懂= =！
- 但是不妨碍我们用= =！

心想事成哦