Union Find

#G面试准备

Number of Islands

Number of Islands - LeetCode

- 写出union find类
- 用到计数 有多少个集合的count
- 先数1的个数 初始化count
- 再对每个1上下左右找1
- 上下左右不仅要满足边界条件 还要是1 不要忘记判断
- 二维转一维是 x * m + y，不是 n；n是行数m是列数

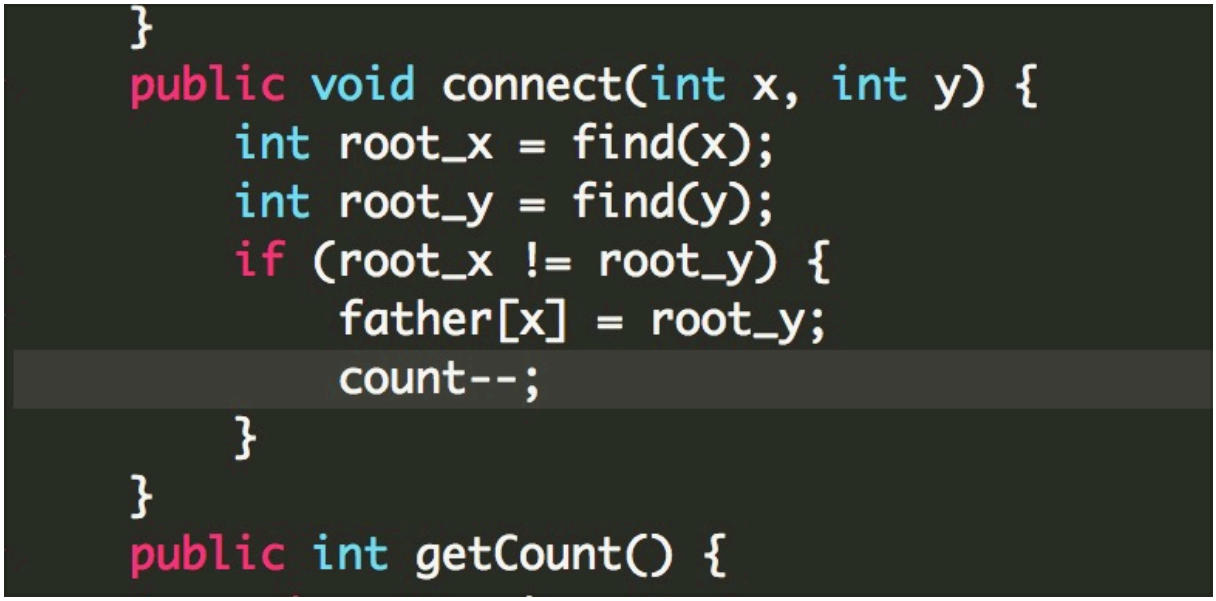
Number of Islands II

Number of Islands II

WA test case:

```
3
3
[[0,1],[1,2],[2,1],[1,0],[0,2],[0,0],[1,1]]
```

错误原因:



- father[root_x] = root_y 才对
- 如果只改了father[x] 并没有根本上改全部的father (其实还是不是很理解 但是一定要记住 要改

的是root的father)

- 还是一个跟上面一样的Union Find问题
- 只不过这回要在每次加入新的点的时候 改count+1
- 然后再上下左右去找是1的点connect connect里面如果不是同一个集合会减1

684. Redundant Connection

Redundant Connection - LeetCode

- 看起来是一道图里找树bad edge的题
- 其实可以用union find做
- 给的是edges数组 遍历edges数组 如果两个节点不是同一个集合 就连接
- 是的话直接输出
- 注意输入的是正好比树多一条边 所以可以初始化unionfind有边数n个entry
- 又因为是从1..n 可以多加一位 把0忽略 比较方便使用

685. Redundant Connection II

Redundant Connection II - LeetCode

- 这道比较像面经里删bad node的题
- 因为tree其实是有方向的 从parent指向child 有两种bad node
  - 一个node有两个parent
  - 有环
  - 以上两种情况同时
- 思路是先看有没有两个parent
  - 如果有两个parent 先挪掉其中一条edge can2 然后做unionfind
    - 如果没有找到环 就说明挪对了 输出挪掉的can2
    - 如果找到了环 那就应该把can2加回来 输出can1
  - 如果没有两个parent 直接输出找到环的时候的最后一条edge
- 找环用unionfind
  - 如果两个点已经在同一个集合 说明这条边在环内
- 找到环的时候 判断有没有两个parent candidates 如果没有就直接输出当前 u → v 如果有 输出那个没有被暂时挪掉的can1
  - 一举两得 又去除了环又保持每个node只有最多一个parent
- 找两个parent其实也算是unionfind 对于边 u → v 如果当前parents[v]已经有了别的值 说明他有两个parent了
  - 此时要暂存两个边can1 can2 并且把其中一条的原edges数组改为0 暂时挪掉