1.

(a)

先在T里loop一遍看新的edge存不存在，如果已经有这个edge了，就比较谁的weight小，把小的那个留下

如果没有这个edge，

同枝：从u开始往上找，找他的父亲，从v开始往上找，找他的父亲，如果能在v path上找到u，或是能在u path上找到v，这两种情况的话都说明uv在同一支，这时把new edge

加进去，T’ = T UNION new edge，必然 形成一个cycle，在这个cycle里找出最重的那一支，删掉就可以

不同枝（同父亲）：如果从u往上找，v往上找，找不到对方，就说明uv现在分别在不同的path上，找这两个path的上的所有edge（路径）。

u path - v path union v path - u path。这个set union new edge，就能组成一个cycle，在这个cycle里找出最重的那一支，删掉就可以

（或者也可以说把两个path里的所有edge行成一个set，把这个set里的所有哦相同的edge都删掉，再把new edge加进去，形成一个cycle）

(b)

总体思想：删掉edge (u, v) 以后，变成了两个tree，找这两个tree，一个tree是跟root连着的，另一个是断开了的。对一个tree的每个节点判断他是否有连接到另一个tree的路径，如果有的话，挑出weight最轻的连起来。

删掉一个edge，有可能在这个tree上，也有可能不在这个tree上

如果不在这个tree上：找不到他，不管他

如果在这个tree上：就说明uv肯定是父子关系（不可能在不同枝，因为如果再不同枝就已经形成了一个cycle了）

如果u是父亲v是孩子，建立两个empty set，copy一下tree T，把u和v分别放进去两个set，分别叫做u set和v set，因为u是父亲，他是断开这个树以后上面那个tree的东西， 那u的父亲，父亲的父亲，u的整条path上的node都往u set里面放，同时把这些edge从T copy里删掉了，对T copy里的每个node找他和他的父亲在不在set u里或set v里，如果不在就继续往上找，总归会找到他的祖先在v set里面（找到v了），那就把这一个path上的node全部加到set v里面，或是找到他的祖先在set u里面（找到u）了，那就把这一个path上的node全部加到set u里面，直到找到T copy里没有剩下的edge，那这时这个MST就被分到了两个不同的set里（两个不同的tree），然后对set Edge 里的每一个edge判断他的两个端点是否一个在set u里一个在set v里，如果不在的话就略过，如果在的话就说明这个edge是连接两个subtree的edge，找出这些edge中weight最小的那个edge连上就好。

2.

tell：

如果我已经有n个人我找过的了，当这两个人其中一个人我找过了，有一个人没找过，unknown

如果都找过，判断是否在一个group里，在的话，看在不在同一个集合，如果在同一个集合，那就是same，如果

add：

如果这两个东西都没找过，要新建一个group，把它们两个放到里面不同的set里

如果这两个东西都找过了，看她俩在不在同一个group，

a)如果在同一个group：

如果在同一个set：

如果不在同一个set：

b)如果不在同一个group：