LCP 5. 发 LeetCoin

难度 困难 6 收藏 分享 切换为英文 关注

题目描述 评论 (13) 题解 (9) New 提交记录

力扣决定给一个刚组团队发 LeetCoin 作为奖励。同时，为了监控给大家发了多少 LeetCoin，力扣有时候也会进行查阅。

该刚组团队的管理模式可以用一棵树表示：

1. 团队只有一个负责人，编号为1。除了该负责人外，每个人有且仅有一个领导（负责人没有领导）；
2. 不存在循环管理的情况，如A管理B，B管理C，C管理A。

力扣想进行的操作有以下三种：

1. 给团队的一个成员（也可以是负责人）发一定数量的 LeetCoin；
2. 给团队的一个成员（也可以是负责人），以及他/她管理的所有人（即他/她的下属，他/她下属的下属，……），发一定数量的 LeetCoin；
3. 查询某一个成员（也可以是负责人），以及他/她管理的所有人被发到的 LeetCoin 之和。

输入：

1. N 表示团队成员的个数（编号为1～N，负责人为1）；
2. leadership 是大小为 (N - 1) * 2的二维数组，其中每个元素 [a, b] 代表 b 是 a 的下属；
3. operations 是一个长度为 Q 的二维数组，代表以时间排序的操作，格式如下：
    1. operations[i][0] = 1：代表第一种操作，operations[i][1] 代表成员的编号，operations[i][2] 代表 LeetCoin 的数量；
    2. operations[i][0] = 2：代表第二种操作，operations[i][1] 代表成员的编号，operations[i][2] 代表 LeetCoin 的数量；
    3. operations[i][0] = 3：代表第三种操作，operations[i][1] 代表成员的编号；

输出：

返回一个数组，数组里是每次查询的返回值（发 LeetCoin 的操作不需要任何返回值）。由于发的 LeetCoin 很多，请把每次查询的结果模 1e9+7（1000000007）。

思路

1　new 一个 coin 数组保存每个人收到的 LeetCoin 数量

2　new 一个 b 数组保存每次查询的数值

3 遍历 operations 数组，operations[i][0]的值对应三种操作

4 其中 operations[i][0]=1，代表第一种操作，当前成员加 LeetCoin 数量

operations[i][0]=2，代表第二种操作，solution 函数作用是将当前成员以及他所管理的下属加 LeetCoin 数量

operations[i][0]=3，代表第三种操作，select 函数的返回值为当前成员以及他的下属成员所拥有的 LeetCoin 的总和，并将 select 的返回值保存在数组 b 中。

5 返回数组 b。

```
1.  class Solution {
2.      public int[] bonus(int n,int[][] leadership,int[][] operations){
3.          int[] coin=new int[n+1];
4.          for(int i=0;i<coin.length;i++){
5.              coin[i]=0;
6.          }
7.          int number=0;
8.          for(int i=0;i<operations.length;i++){
9.              if(operations[i][0]==3){
10.                 number++;
11.             }
12.         }
13.         int[] b=new int[number];
```

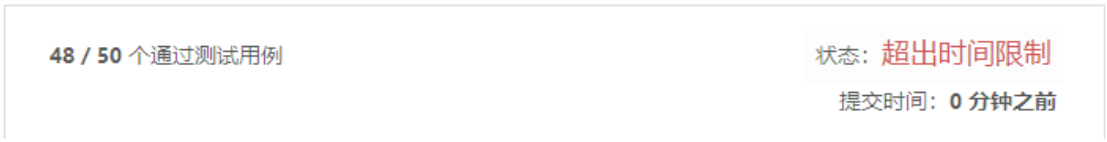
```
        int k=0;
        for(int i=0;i<operations.length;i++){
            if(operations[i][0]==1){
                int index=operations[i][1];
                coin[index]=coin[index]+operations[i][2];
            }
            if(operations[i][0]==2){
                int index=operations[i][1];
                //coin[index]=coin[index]+operations[i][2];
                solution(index,operations[i][2],leadership,coin);
            }
            if(operations[i][0]==3){
                int count=select(operations[i][1],coin,leadership);
                b[k++]=count;
            }
        }
        return b;
    }

    private int select(int index, int[] coin, int[][] leadership) {
        // TODO Auto-generated method stub
        int total=0;
        for(int j=0;j<leadership.length;j++){
            if(leadership[j][0]==index){
                total=total+select(leadership[j][1],coin,leadership);
            }
        }
        total=total+coin[index];
        return total;

    }
    private void solution(int index, int i, int[][] leadership, int[] coin)
{
        // TODO Auto-generated method stub
        for(int j=0;j<leadership.length;j++){
            if(leadership[j][0]==index){
                solution(leadership[j][1],i,leadership,coin);
            }

        }
        coin[index]=coin[index]+i;
    }
}
```

提交记录



缺点：for 循环以及递归的调用，导致时间复杂度太高了