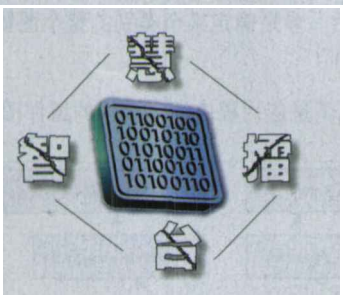

称球问题的解决方法

□ 文／司德谭

> 数学问题和编程问题往往都需要严谨的逻辑、反复的推敲，数学问题往往不只是提出来难为人们的脑筋的，实际上很多数学问题最终转化成为实际应用的模型。这篇文章为这个问题的编程解决开了个好头，或许也能为我们的思路拓展开一个头。

称球问题，是指在N个球中有一个球重量和其他N-1个球稍有不同(或轻或重)，在没有砝码的天平上，经过k次的称量比较，找出这个异重的球，并能够知道是比其他球重，还是轻。限制次数k和球数N之间的关系是：N <= (3^k-1)/2。

关于计算过程的数学求证，在题目中提供的链接上可以找到，这里就不再重复了。在这儿就分析一下用程序如何实现这个判断的过程。

先分析一下，解决这个问题的思考过程，每个球有3种可能：标准重量，重量偏轻，重量偏重。3种可能组合产生4种状态：已知为标准球；已知为标准球或偏轻球(不可能偏重)；已知为标准球或偏重球(不可能偏轻)；3种可能都有（可能为标准球，也可能偏轻或偏重)。在初始状态下，N个球都是处于第4种状态的。

开始动手的时候，需要先把N个球分为3组。可是，为什么要分为3组呢？天平可只有两个托盘啊！托盘固然只有两个，可是如果把一部分球(就叫“旁置球”吧)不放在天平上，通过天平的称量结果，也可以推断出旁置球的部分信息来。比如说，把天平一部分球（就叫“右侧球”）放在天平的右侧托盘，一部分球（就叫“左侧球”）放在天平的左侧托盘，如果天平左右不平，就可以推断异常球在左侧球或右侧球中，而旁置球全部是标准球；如果天平左右平衡，就可以推断异常球在旁置球中，而左侧球和右侧球都是标准球。那么，分为3组的球数，又该如何怎么定呢？原则就是让3组球数尽可能的平均。可是如果不能平均分配怎么办，即N不是3的倍数，那么分为（N/3+1)的两组，然后其余的作为第三组。用n表示前两组的球个数，则：n=(p_unknowNum%3)?(p_unknowNum/3)+1:(p_unknowNum/3)。

所有球编号为0至N-1，经过分组后，第一组球为0至n-1，第二组为n至(2*n-1)，第三组为(2*n)至N-1。把第一组放入天平右侧，把第二组放入天平左侧，把第三组旁置。然后开始称量。称量可能产生三种结果：右侧重，左侧重，左右等重。如果右侧重，则说明第三组都是标准球，异常球在第一二组中，且第一组球只有标准球或偏重球的可能，第二组球只有标准球或偏轻球的可能。如果左侧重：则说明第三组都是标准球，异常球在第一二组中，且第一组球只有标准球或偏轻球的可能，第二组球只有标准球或偏重球的可能。如果左右等重，则说明异常球在第三组中，第一二组球都是标准球。

经过一轮称重，虽然还不能确定那个球是异常球，但是毕竟范围缩小了不少。开始进入第二轮称重。如果第一轮的称重结果是左右等重，那么让N等于N-2*n，范围缩小为第一轮分组产生的第三组中，重复第一轮的过程。如果第一轮的称重结果不是左右等重，则对第一组和第二组的球进行调整。已知第一二组中异常球，共有2*n个。让N等于2*n，再次按前边计算分组球数的方法，计算m=(p_unknowNum%3)?(p_unknowNum/3)+1:(p_unknowNum/3)。然后从第二组中移出m个未定球作为第三组，从第一组中移m个未定球到第二组中，从已知的标准球中任选m个放到第一组中，再次组成三组球（从第一组中移到第二组的m个未定球，称为移动球)。然后开始第二轮称重。称量同样可能产生三种结果：右侧重，左侧重，左右等重。如果左右等重，则表示异常球在第三组中，而且可以推定异常球状态为偏轻，其他未定球都为标准球。如果不是左右等重，且本次称量结果重的一侧和前一轮一样，则说明异常

球在本次分组时没有移动的球中，而第三组球和移动球都是标准球；如果称量结果重的一侧和前一轮不一样，则说明异常球在移动球中，而第三组球和没有移动的球都是标准球；并且，如果本次右侧重就可以推算异常球为偏轻，如果左侧重就可以推算异常球为重。

如果在上一轮后，还不能确定异常球，则进行下一轮称量准备。如果上一轮的结论是异常球在旁置球(第三组)中，或是在移动球中，则按第一轮分组方式进行；如果是在未动球中，则继续按第二轮的方式继续进行。

在未定的球数很少的时候，可能通过球的状态，异常球的性质就可能推断出来是哪个球来。这个细节过程，读者朋友可以自己推算一下。

从以上过程描述可以看出，在程序实现中，可以有条件地使用递归处理来解决。

为了实现程序处理，需要将相关信息用数字表示：

```
    char weightStatus[][9]={"未知","偏轻","偏重"};
    // 异常球可能状态
    char metageStatus[][9]={"未知","右侧重","左右等重","
左侧重"}; // 天平称量可能结果
    char BallStatus[][13]={"标准球","标准或轻","标准或重",
"标准或轻或重"}; // 球可能状态
```

程序中每次称量处理，都会产生大量的环境信息，如每个球的状态信息，球在天平上的位置信息，称量的结果，对异常球的定性等。这些信息需要传递到下一轮称量处理中，并对称量结果产生影响。为了方便把这些信息组织起来，便于维护，定义一个类 CMetage 来存取这些信息：

```
class CMetage{
public:
        CMetage(void);
        ~CMetage(void);
        int putRight(const int* p_rightList, int p_listNum);
        int putLeft(const int* p_leftList, int p_listNum);
        int putMove(const int* p_moveList, int p_moveNum);
        int putOther(const int* p_otherList, int p_otherNum);
        int putBallStatus(const int* p_ballStatus, int
                          p_ballNum);
        const int* getRightList();
        const int* getLeftList();
        const int* getMoveList();
        const int* getOtherList();
        int getBallStatus(int* p_ballStatus, int* p_ballNum);
        int getUnknowList(int* p_unknowList, int*
                          p_unknowNum);
        int getNormalList(int* p_normalList, int*
                          p_normalNum);
        int getRightNum();
        int getLeftNum();
        int getMoveNum();
        int getOtherNum();
        int getBallNum();
        int getUnknowNum();
        int getNormalNum();
        int setStatus(int p_mStatus, int p_mbStatus=0);
        int getStatus(void);
        int setwStatus(int p_wStatus);
        int getwStatus(void);
        int getsStatus(void);
private:
        int m_ballStatus[MAXBALL];
        int m_ballNum;
        int m_rightList[MAXBALL];
        int m_rightNum;
        int m_leftList[MAXBALL];
        int m_leftNum;
        int m_moveList[MAXBALL];
        int m_moveNum;
        int m_otherList[MAXBALL];
        int m_otherNum;
        int m_status;
        int m_wStatus;
        int m_sStatus;
};
```

递归处理的主函数，需要根据输入的待出来的未定球列表，以及上一轮的称量状态信息，来决定本次处理：

```
int findBall(
    const int* p_ballStatus, // 指向存储所有小球状态的数组指针
    int p_ballNum,           // 表示小球状态的数组中的有效个数
    const int* p_unknowList,// 指向存储所有未定球编号的数组的指针
    int p_unknowNum,         // 表示未定球编号数组中的有效个数
    const int* p_normalList, // 指向存储所有标准球编号的数组的指针
    int p_normalNum,         // 表示标准球编号数组中的有效个数
    CMetage* p_m,            //上次称量的状态对象指针，初始为NULL
    int p_mNum               // 当前为第几次称量
);
```

按上边的处理思路，模拟一次称量的过程，小球的状态变化如图 1 所示。

	0	1	2	3	4
0	3	2	2	2	0
1	3	2	2	2	2
2	3	2	2	2	0
3	3	2	2	0	0
4	3	2	0	0	0
5	3	1	0	0	0
6	3	1	0	0	0
7	3	1	0	0	0
8	3	1	0	0	0
9	3	1	0	0	0
10	3	0	0	0	0
11	3	0	0	0	0
12	3	0	0	0	0
13	3	0	0	0	0

图 1

图 2

图示中是一次具体称量过程，在实际实现中，数值会因天平称量结果和小球排放位置不同而变化。程序执行的输出效果大致如 2 所示。

有网友在站点上列出了自己的思路，但很可惜不太完整。上边也有其他相关的资料和思路，可供大家再作谈论。

除了这种穷举所有可能的版本外，这个问题还可以做一个模拟自动查找的版本。大致思路是，提供一个小球重量的列表，其中有一个小球的重量和其他小球略有差异。这个重量的列表规定为只有天平类的称量方法中可以访问，用程序只借助这个天平类把异常球从列表中找出来。有兴趣的朋友，不妨自己尝试一下。■

■ 责任编辑：胡心庭 (huxt@csdn.net)