圆圈中最后剩下的数\_剑指Offer\_62

# 圆圈中最后剩下的数\_剑指Offer\_62

## 题目描述：

\* **难度：Medium**

每年六一儿童节,牛客都会准备一些小礼物去看望孤儿院的小朋友,今年亦是如此。HF作为牛客的资深元老,自然也准备了一些小游戏。其中,有个游戏是这样的:首先,让小朋友们围成一个大圈。然后,**他随机指定一个数m**,让编号为0的小朋友开始报数。每次喊到m-1的那个小朋友要出列唱首歌,然后可以在礼品箱中任意的挑选礼物,并且不再回到圈中,从他的下一个小朋友开始,继续0...m-1报数....这样下去....直到剩下最后一个小朋友,可以不用表演,并且拿到牛客名贵的“名侦探柯南”典藏版(名额有限哦!!^\_^)。请你试着想下,哪个小朋友会得到这份礼品呢？(注：小朋友的编号是从0到n-1)

## 思路分析

\* 思路: 方法1：构建环形链表模型模拟圆圈，主要是实现节点删除功能。

\* 方法2：探寻数学规律，利用公式直接计算。

\* 每次删除一个点之后，都映射到新的顺序，如0--n-1,删除m后，

\* 映射到0--n-2；然后再映射到0--n-3等等。

\* 先是n从大到小，最后到1，此时有个正映射关系，

\* 根据反函数，再利用逆映射，有n=1逐渐映射到n。

\* n从大到小映射关系为：p(x)=[x-m]%n;

\* 反过来，n从小到大映射关系为:p-1(x)=[x+m]%n;

\* 因此，f(n,m)=p-1[f(n-1,m)]=[f(n-1,m)+m]%n;

\* 最后的规律：f(n,m)=[f(n-1,m)+m]%n;当n=1时，f(n,m)=0;

\* 利用递归或循环都可以实现。

\* 具体推导过程：见《剑指Offer\_P300》。

## Java代码

### 方法1：环形链表

\* \*\*\*\*\*\*\*\*\*方法1：利用环形链表实现。\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public int **LastRemaining\_Solution**(int n, int m) {

if( n < 1|| m < 0) return -1;

if( n == 1||m == 0) return 0;

ListNode start = constructCircularLinkedList(n);

int len = n;

while(len > 1 ){

int step = (m-1)%len;//删除第m个节点，故走m-1步

for(int i = 1;i <= step;i++){

start = start.next;

}

//删除start节点<利用伪删除>

start.val = start.next.val;

start.next = start.next.next;

len--;

}

return start.val;

}

//共用的链表结构ListNode

private class ListNode {

int val;

ListNode next = null;

ListNode(int val) {

this.val = val;

}

}

//构建0-(n-1)的环形链表

public ListNode **constructCircularLinkedList**(int n){

ListNode start = new ListNode(0);

ListNode current = start;

for(int i = 1;i < n;i++){

current.next = new ListNode(i);

current = current.next;

}

current.next = start;

return start;

}

### 方法2：数学规律

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*方法2：数学公式\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* **f(n,m)=[f(n-1,m)+m]%n; 当n=1时，f(n,m)=0;**

\*/

//循环实现

public int LastRemaining\_Solution2(int n, int m) {

if(n < 1|| m < 0) return -1;

if(n == 1||m==0) return 0;

int last = 0;

for(int i = 2;i <= n;i++){

last = (last + m)%i;

}

return last;

}

//递归实现

public int LastRemaining\_Solution3(int n, int m) {

if(n < 1|| m < 0) return -1;

if(n == 1||m==0) return 0;

return (**LastRemaining\_Solution3(n-1,m)**+m)%n;

}