复杂链表的复制\_剑指Offer\_35

# 复杂链表的复制\_剑指Offer\_35

## 题目描述：

\* 输入一个复杂链表（每个节点中有节点值，以及两个指针，

\* 一个指向下一个节点，另一个特殊指针指向任意一个节点），

\* 返回结果为复制后复杂链表的head。

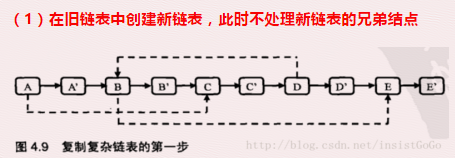
\* （注意，输出结果中请不要返回参数中的节点引用，否则判题程序会直接返回空）

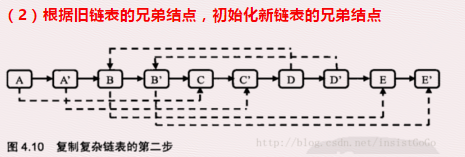
## 思路分析：

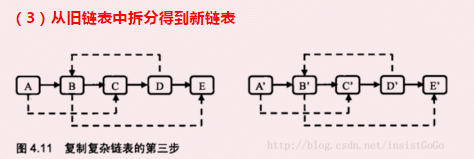
\* 方法：分成三步走：第一步：首先依次复制新节点如A的A'，建立实链接A->A'->B->B'；

\* 第二步：依次对复制节点复制对应的虚链接；

\* 第三步：拆分链表，奇数节点就是原来的链表，偶数节点即为复制链表；







## Java代码

### 一个步骤一个方法

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*实现1：一个步骤对应一个方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public RandomListNode Clone(RandomListNode pHead){

firstStepOfCopyNode(pHead);

secondStepOfCopyRandomLinked(pHead);

return thirdStepOfSplitLinkedList2(pHead);

}

/\*\*

\* 第一步：复制节点，建立实链接。A->A'->B->B'->C->C'

\*/

public void firstStepOfCopyNode(RandomListNode head){

RandomListNode originalNode = head;

while(originalNode != null){

RandomListNode copyNode = new RandomListNode(originalNode.label);

copyNode.next = originalNode.next;

originalNode.next = copyNode;

originalNode = copyNode.next;

}

}

/\*\*

\* 第二步：复制虚链接.

\*/

public void secondStepOfCopyRandomLinked(RandomListNode pHead){

RandomListNode originalNode = pHead;

while(originalNode != null){

RandomListNode copyNode = originalNode.next;

if(originalNode.random != null)

copyNode.random = originalNode.random.next;

originalNode = copyNode.next;

}

}

/\*\*

\* 第三步：拆分链表：奇数节点为原链表；偶数节点为复制链表。

\*

\* 这个方法需要注意:在拆分最后，原链表的最后一个节点的next需要指向null；

\* 容易出现：原链表为A->B->C->C'的错误情况。

\*/

public RandomListNode thirdStepOfSplitLinkedList(RandomListNode pHead){

if(pHead == null) return null;

//初始化

RandomListNode originalNode = pHead;

RandomListNode copyHead = pHead.next;

RandomListNode copyNode = copyHead;

originalNode.next = copyNode.next;

originalNode = originalNode.next;

//继续循环处理后续节点

while(originalNode != null){

copyNode.next = originalNode.next;

copyNode = copyNode.next;

originalNode.next = copyNode.next;

originalNode = originalNode.next;

}

return copyHead;

}

/\*\*

\* 第三步的第二种实现方式

\*/

public RandomListNode thirdStepOfSplitLinkedList2(RandomListNode pHead){

if(pHead == null) return null;

//初始化

RandomListNode originalNode = pHead;

RandomListNode copyHead = pHead.next;

RandomListNode copyNode = copyHead;

//继续循环处理后续节点

while(copyNode.next != null){//下一个原节点不为空

originalNode.next = copyNode.next;

originalNode = originalNode.next;

copyNode.next = originalNode.next;

copyNode = copyNode.next;

}

originalNode.next = null;//手动指定原链表末节点next为null

return copyHead;

}

### 综合在一起的方法

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*综合在一起的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public RandomListNode Clone2(RandomListNode pHead){

if(pHead == null) return null;

RandomListNode originalNode = pHead;

RandomListNode copyNode = null;

**//第一步：复制节点，构造成A->A'->B->B'->C->C'**

while(originalNode != null){

copyNode = new RandomListNode(originalNode.label);

copyNode.next = originalNode.next;

originalNode.next = copyNode;

originalNode = copyNode.next;

}

**//第二步：复制虚链接**

originalNode = pHead;

while(originalNode != null){

copyNode = originalNode.next;

if(originalNode.random != null)

copyNode.random = originalNode.random.next;

originalNode = copyNode.next;

}

**//第三步：拆分链表，奇数为原链表，偶数为新复制链表**

originalNode = pHead;

RandomListNode copyHead = pHead.next;

copyNode = pHead.next;

while(copyNode.next != null){

originalNode.next = copyNode.next;

originalNode = originalNode.next;

copyNode.next = originalNode.next;

copyNode = copyNode.next;

}

**originalNode.next = null;**

return copyHead;

}