今天来说说关于HashMap的一些事情。大家可以从Leetcode上面找到一到叫做[Copy List with Random Pointer](https://leetcode.com/problems/copy-list-with-random-pointer/) 的题目，今天就拿他开刀。

首先，于我个人而言，各种基本数据结构的操作里面，最令我头大的就是各种list，原因呢可以归结为我们只能从当前节点一股脑的往后找，但是绝对不能回头，无形中增添了好多麻烦比如排序啥的（这也是一道题，也有神一样的方法，以后再说）。

对于一部分同学呢，可能HashMap也是令人头大的东西。这个东西好用是好用毕竟search时候O(1)的神器，（想必大家也都在有时间复杂度要求的题里面想到过这货）但是华丽丽的Key - Value pair （键值对）确实需要腾出一些脑筋去应付。

可是！XiKaXi(此处为日语音译)！想要作为一名合格的coder怎能怕了这两个小家伙。让我们从这道题开始慢慢了解HashMap 和 List吧！

----------------------------------------- 我是分割线， 以下正文--------------------------------------------

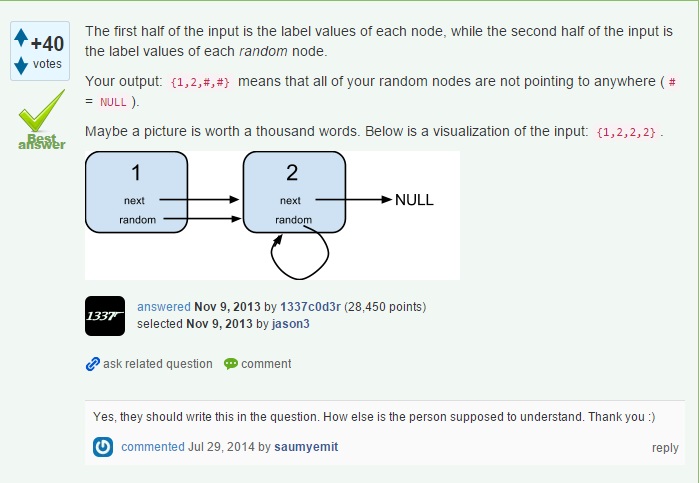
（是的前面都是废话，这句也是）

首先，国际惯例，题目介绍：

A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a deep copy of the list.

作为Leetcode上面的hard题型，只用两句话就描述完了全题的行径，果然不负众望的让全场没有一个人读懂他究竟想干什么。于是，leetcode老大在人们不断的谴责声中站了出来，给出了官方解释：



大意就是：首先我们有一个常规的LinkedList，接着我们向每一个LinkedList的Node里面加入一个叫做random 的pointer，随机指向一个list中的node （当然也可以是本身）。于是乎，如上图所示，每个node都有next 和 random两个指针。关于输出结果，其实只要复制的正确，大可无视，想知道的话上图解释的也很完美，不再赘述。

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

现在，开始思考。

假设，你完全没有开到开篇的废话，也就是没有人暗示你可以使用HashMap，你会如何解答这道题？

首先，题目要求复制一个LinkedList，大部分同学想必都做过复制普通LinkedList的事。根据原来的Node创建新Node并且依次链接起来，返回新list的head，大功告成！接下来就是在新List上面继续完成补充random指针的活儿了。

这是一定可行的，而且也是一定正确的。

问题在于：当我们想要补充random指针的时候，因为他不再是连续的，即存在这两个（反正我只想到了两个，如有更多望补充）很麻烦的问题。一：我们没法保证通过这些random指针可以访问到每个节点，也就是说，我们还需要额外的步骤来确定我们已经为每一个Node赋予了新的random指针。二：原有的random指针如字面意思一样指向随机的位置，而这个位置我们并不知道他的absolute index，我们并不能简单的找到新list里面的对应的Node。

麻烦了对不对？

所以，我当机立断的舍弃了这种铁定不是最优解的想法。转而进行下一步思考。

如你们所料，我真的啥也没想出来……

果断的翻开discuss，找到了点赞最多的java解，认真进行了拜读。然后服了。

关键字： HashMap

尝试思考一下，在得到HashMap的hint之后，你有什么思路？

没有？太正常了，我也没有……

（有的无视以上这句）

退一步想，既然用HashMap，必然要确定Key - Value的类型，你会怎么定义？

无外乎：<Integer,Node> <Integer,Node> <Node,Node>这三种嘛……

第一种<Integer,Node> 似乎多用于查找Node，通过label确定Node，貌似没法用在创建新的List，更别提复制……难道要把整个map遍历一遍去复制？不靠谱，pass

第二种：<Integer,Node>同理 很难建立起旧List和新List间的联系，pass

第三种：……总要选一个吧，而且两个Node确实隐约存在真新旧两List间千丝万缕的联系。

接下来的事情，就是小菜鸡和大粗腿之间的差距了（请坚信这个鸿沟可以通过刷题弥补！）

选定了<Node,Node>类型后要确认放入的分别是什么！

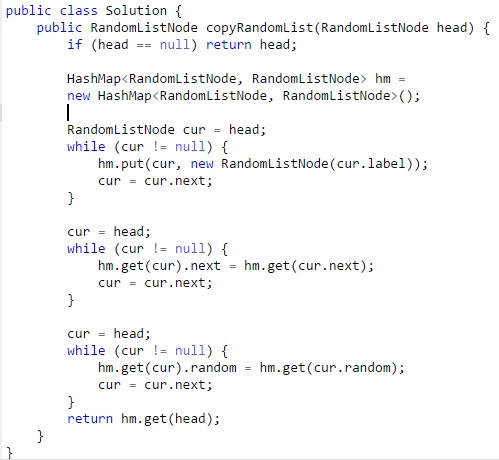
（我是用Java刷的^ . ^）(不要在意语法，我只是为了好说明白)

HashMap<oldNode, new Node(oldNode.label)> hm = new HashMap<oldNode, new Node(oldNode.label)>();

了解到这里，我承认当时的我脑子里飞过了一万个灵感，但是一想到是LinkedList还是没耐着性子思考下去，继续看了大神的解。

不过现在，时间充裕了，不妨思索一下。知道了键值对的设定，你有什么想法么。旧节点和新节点存放在一起，每当找到一个Key得到的是新List的Node作为Value，简直太方便了对不！对旧节点的操作此时全等于对新节点的操作！

后面的就是体力活了。不再多扯，上代码：



事后话：

Discuss区大神云集，其他的解也有很精妙的，但均比不上此法带给我的震撼和欢愉。

代码简洁，思路清晰，方法巧妙，说实话我要是面试官我肯定想看到这个。太舒服。

没谁了，真的。