BTC的数据结构

hash指针：

- 指针内容不仅包含地址，还有地址块内容的hash值，寻址的同时还可以确认内容是否被篡改。
- 比特币中最基本的结构就是区块链，区块链就是一个一个区块组成的链表。区块链和普通的链表相比有什么区别:

    1. 用哈希指针代替了普通指针(B block chain is a linked list using hash pointers) 区块链第一个区块叫作创世纪块(genesis block) 最后一个区块 是最近产生的区块(most recent block) 每一个区块都包含指向前一个区块的哈希指针
    一个区块的哈希指针怎么算:是把前面整个区块的内容，包括里面的hash pointer ，合在一起取哈希值。通过这种结构，可以实现tamper-evident log。如果有人改变了一个区块的内容，后面一个区块的哈希指针就对不上，因为后一个区块哈希指针是根据前一个区块的内容算出来的，所以后一个哈希指针也得改，以此类推，我们保留的是最后一个哈希值也会变化。
    2. 普通链表可以改变任意一个元素，对链表中其他元素是没有影响的。而区块链是牵一发而动全身，因为只需要保存最后一个哈希值，就可以判断区块链有没有改变，在哪里改变了。

因此比特币没有要保存所有区块的内容，可以只保留最近的几千个区块。如果要用到以前的区块，可以向系统中其他节点要这个区块。有些节点是有恶意的，怎么判断?这里要用到哈希值一个性质，如下: 其他节点给你一个区块，如何判断它是正确的?算出它的哈希值，与保留的区块的哈希值对比，即可。

Merkle tree

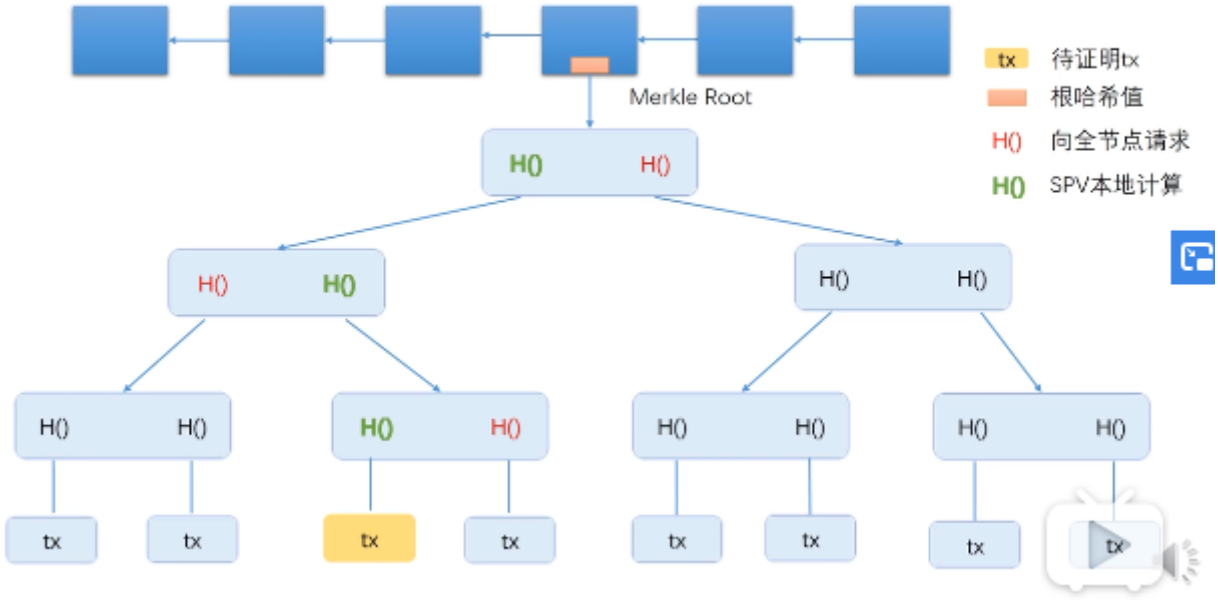
比特币中的另外一个结构是:Merkle tree。(见拍的图②，其中最下面一层是数据块(data blocks)，上面三层内部节点都是哈希指针(hash pointers)，第一层是根节点，根节点的区块也可以取个哈希，叫根哈希(root hash)) 另外一个概念:binary tree。Merkle tree就是指针的哈希指针的binary tree

这种结构的好处:只要记住根哈希值，就能检测出对树中任何部位的修改。

比特币当中各区块之间用哈希指针连接在一起，每个区块所包含的交易组织成一个merkle tree的形式，最下面一行data blocks每个区块实际上是一个交易，每个区块分为两部分，分别是块头和块身(block header ,block body)。块头里面有根哈希值，每个区块所包含的所有交易组成的merkle tree的根哈希值存在于区块的块头里面，但是，块头里没有交易的具体内容，只有一个根哈希值，块身里面是有交易的列表的。

merkle tree 的作用:①提供merkle proof 比特币中的节点分为两类:全节点(保存整个区块的内容，即块头块身都有，有交易的具体信息)和轻节点(例如手机上的比特币钱包)(只有块头)

这时存在一个问题:如何向一个轻节点证明某个交易是写入区块链的? 这时需要用到merkle proof :找到交易所在的位置(最底行的其中一个区块)，这时该区块一直往上到根节点的路径就叫merkle proof。



最上面一行是小型的区块链，该图展现的是一个区块的merkle tree，最下面一行是包含的交易。假设某个轻节点想知道图中黄色的交易，是否包含在了merkle tree里面。该轻节点没有包含交易列表，没有这颗merkle tree的具体内容，只有一个根哈希值。这时轻节点向一个全节点发出请求，请求证明黄色的交易被包含在这颗merkle tree里面的merkle proof。全节点收到这个请求之后，只需要将图中标为红色的这三个哈希值发给轻节点即可。有了这些哈希值之后，轻节点可以在本地计算出图中标为绿色三个哈希值。首先算出黄色交易的哈希值，即它正上方的那个绿的哈希值，然后跟旁边红色的哈希值拼接起来，可以算出上层节点绿色的哈希值。然后再拼接，再算出上层绿色哈希值，再拼接，就可以算出整棵树的根哈希值。轻节点把这个根哈希值和block header里的根哈希值比较一下，就能知道黄色的交易是否在这颗merkle tree里。

全节点在merkle proof里提供的这几个哈希值，就是从黄色的交易所在的节点的位置到树根的路径上用到的这些哈希值。轻节点收到这样一个merkle proof之后，只要从下往上验证，沿途的哈希值都是正确的即可。(验证时只能验证该路径的哈希值，其他路径是验证不了的，即该图中红色的哈希值是验证不了的)

这样是否不安全呢?假如黄色交易被篡改，它的哈希值发生了变化，那能不能调整旁边红色的哈希值，使得它们拼接起来的哈希值是不变的呢?不行，根据collision resistance，这是不可行的。 merkle proof可以证明merkle tree里面包含了某个交易，所以这种证明又叫proof of membership或 proof of inclusion。
对于一个轻节点来说，验证一个merkle proof 复杂度是多少?假设最底层有n个交易，则merkle proof 复杂程度是**θ(log(n))**

如何证明merkle tree里面没有包含某个交易?即proof of non-membership。可以把整棵树传给轻节点，轻节点收到后验证树的构造都是对的，每一层用到的哈希值都是正确的，说明树里只有这些叶节点，要找的交易不在里面，就证明了**proof of non-membership**。问题在于，它的复杂度是线性的θ(n)，是比较笨的方法。

如果对叶节点的排列顺序做一些要求，比如按照交易的哈希值排序。每一个叶节点都是一次交易，对交易的内容取一次哈希，按照哈希值从小到大排列。要查的交易先算出一个哈希值，看看如果它在里面该是哪个位置。比如说在第三个第四个之间，这时提供的proof是第三个第四个叶节点都要往上到根节点。如果其中哈希值都是正确的，最后根节点算出的哈希值也是没有被改过的，说明第三、四个节点在原来的merkle tree里面，确实是相邻的点。要找的交易如果存在的话，应该在这两个节点中间。但是它没有出现，所以就不存在。其复杂度也是log形式，代价是要排序。排好序的叫作**sorted merkle tree**。比特币中没有用到这种排好序的merkle tree，因为比特币中不需要做不存在证明。

这节讲了比特币中两种最基本的结构:区块链和merkle tree，都是用哈希指针来构造的。除了这两种之外，哈希指针还能用另一个方面。

只要一个数据结构是无环的(非循环链表)，都能用哈希指针代替普通指针。有环的话存在一个问题，他们的哈希值没法计算，没法确定一个哈希值固定的区块。