03 比特币的数据结构

哈希指针

普通的指针存储的是一个结构体的内存地址

哈希指针除了存储结构体的内存地址外，还需要存储结构题的哈希值。不仅可以得到结构体的位置，还可以检测内容是否被篡改过。

区块链就是一个使用哈希指针的链表。Block Chain is a linked list using hash pointers

第一个区块是系统生成的，被称之为genesis block，创世纪块

最后的区块是最近的区块the most recent block

每一个区块都包含前一个区块的哈希指针。

系统里面还保存着最后一个区块的哈希指针。

后一个区块的哈希指针中的哈希值，是前一个区块的内容(包含前一个区块中存储前前一个区块的哈希指针)，这样的结构称为tamper evident log

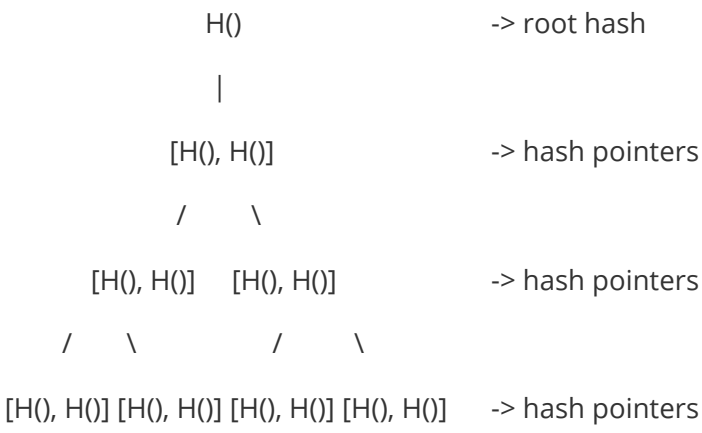
这样设计的好处在于当其中一个区块的内容被篡改了之后，会和其后面区块中存储的哈希值匹配不上，所以必须篡改一系列的内容才可以。所以我们只需要通过系统里存储的最后一个区块的哈希指针中的哈希值，就可以检测出来区块链是否被篡改了。和普通的linked list的区别在于，普通的链表对一个部分进行修改之后，不会影响其他部分，但是区块链当对于其中一个区块的内容进行篡改之后，后影响其余的区块。牵一发而动全身。只需要保存最后一个哈希值，就可以知道是否篡改了之前的区块。

所以每一个节点，只需要保存固定数量的区块的哈希值就可以了，不需要所有的都存。

但是有些节点是恶意的，我们可以通过哈希指针中的存储来验证收到的区块链信息是否被篡改过。

Merkle Tree

用哈希指针替代了binary tree中的普通指针



只需要记住根哈希，就可以检测整棵树是否被篡改了

■（data blocks）实际上是transaction，不同的交易

每个区块包含两个部分，block header和block body.

在block header中，有root hash，但是没有交易的具体内容

在block body中，有交易内容的具体列表

全节点是有block header和block body两部分

轻节点是只有block header

e.g. 比特币手机钱包只使用轻节点，当别人给我转了一笔钱的时候，告诉我交易已经写到了对应的区块当中，如果我只有轻节点（只有root hash），那么我如何验证交易是否存在呢？-> 需要转账给我的人提供一个Merkle proof给我。从root到对应区块的唯一路径，被称之为Merkle proof.

Collision Resistance的性质保证，我们人为篡改交易信息之后（导致算出来的哈希值会变化），我们无法通过篡改从全节点处收到的哈希值，来保证整个节点的哈希不变。因为这样的操作需要我们找到如何篡改红色H()的办法来保证整体的绿色H()加上红色H()不变。这属于人为制造哈希碰撞的情况。Collision Resistance可以保证我们没有高效的办法来篡改。

Merkle proof可以证明某个merkle tree里包含了一个具体的交易。所以merkle proof也叫做proof of membership或者proof of inclusion。

Merkle proof的时间复杂度是O(logn)，其中n是叶节点的数量，也就是区块的数量：我认为的原因是我们知道哈希函数计算的时间复杂度是O(1)，那么对于一个叶节点(leaf node)来说，从leaf node走到root node的最大长度是logn，所以我们最多只需要logn * O(1)这么多次的哈希函数计算来验证，所以时间复杂度是O(logn)

但是如何证明non-membership呢?

最笨的办法就是将整棵树发送给轻节点，然后一一来遍历所有的区块。

但如果我们对叶节点的排列顺序做一些要求，比如说按照交易的哈希值来从小到大排列。

排序的我们称之为Sorted Merkle Tree. 当我们需要证明一个交易不存在的时候，我只需要先算出来这个交易的哈希值。然后在排序好的区块中，找到这个交易的左侧和右侧区块，然后利用哈希函数向上计算（同理于上面的Merkle Proof），最终得出来的根哈希如果和轻节点中存储的值一样，则说明本身的左侧和右侧的两个区块是相邻的（两个区块之间不存在其他的区块），那么我们验证的交易不存在于这个Sorted Merkle Tree中。

有环的的数据结构不能够使用哈希指针，因为会形成循环依赖。没有环的数据结构都可以使用哈希指针。