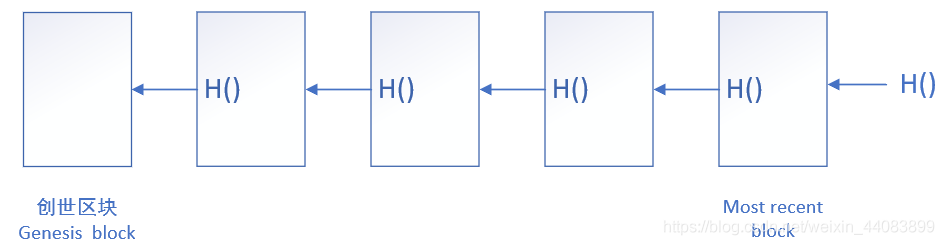
1. **哈希指针**

**1）普通指针P**：存储某个[结构体](https://so.csdn.net/so/search?q=%E7%BB%93%E6%9E%84%E4%BD%93&spm=1001.2101.3001.7020" \t "_blank)在内存中的地址(起始位置)

**2）哈希指针H()**：既存储某个结构体在内存中的地址，也存储对结构体内容的哈希值。好处在于不仅可以找到结构体的位置，还可以检测结构体中的内容有没有被篡改。

**3）区块链与普通链表的区别**：使用哈希指针代替了普通的指针。换句话说，**区块链就是用哈希指针代替普通指针的链表**

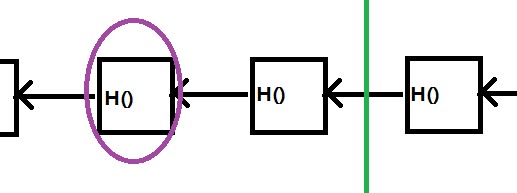


注意：取哈希是将前一个区块的所有内容(实际上是block [header](https://so.csdn.net/so/search?q=header&spm=1001.2101.3001.7020))取哈希，包括前一个区块的哈希指针。最后一个最新产生的区块(Most recent bolck)的哈希指针H()被整个系统的所有节点所存储。

**这种结构的作用**：可以建立一个防篡改日志(tamper-evident log)。系统的多数节点只需要记录最后一个区块(Most recent bolck)的哈希值，前面任意一个区块被篡改都能被检测出来，因为修改了某个区块的内容，会导致它与后面区块保存的哈希值对不上了，那个后面区块的哈希值就也要改，然后再往后的就也要改。类似于多米诺骨牌效应，**前面的修改一定会通过哈希指针传递到最后一个区块**。

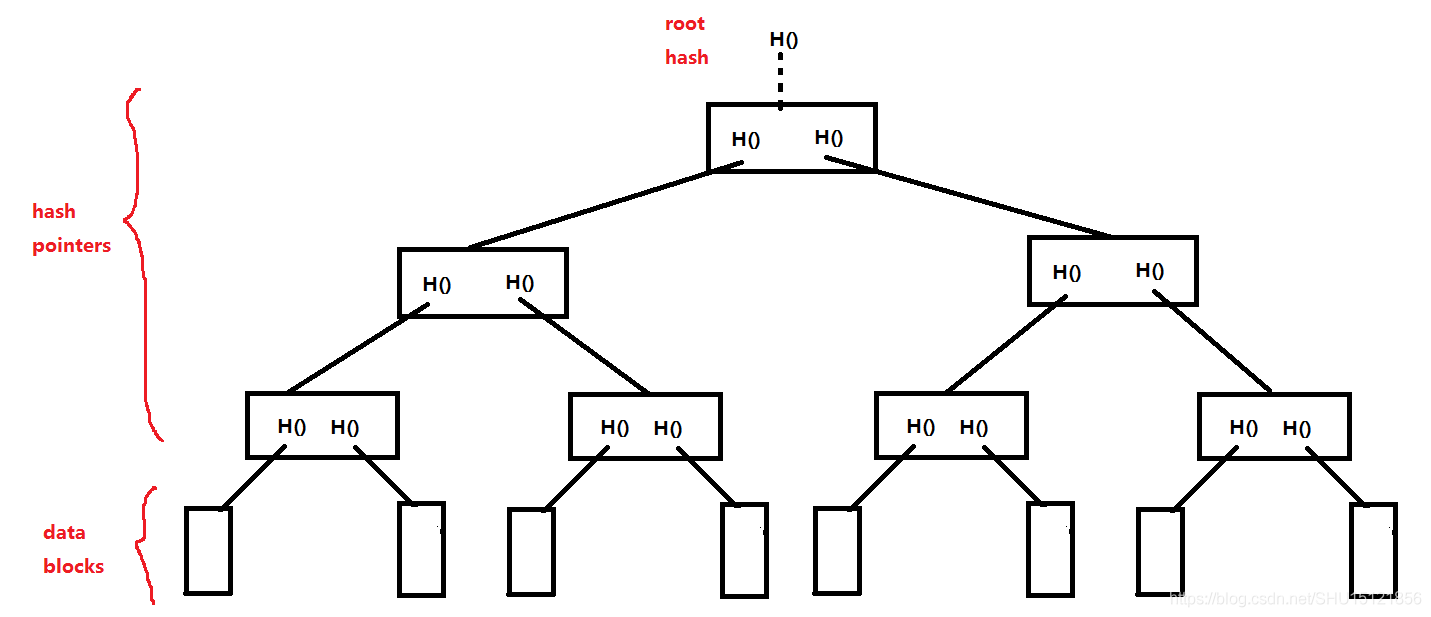
有了这个性质，某一个用户也就没必要保存系统中的所有区块了，可以只保存最近的一些区块，如果要用到先前产生的区块，再向别人要就可以。

比如只保存了下图中绿线右侧的区块，这些是比较新产生的区块，然后如果要用到紫色圈圈出来的区块，可以向别人要。保存的哈希指针在这里可以校验给的区块是不是正确的（取个哈希值然后和保存的哈希指针的哈希值比较一下）。



1. **Merkle tree(默克尔树)**

**Merkle tree本质上是用哈希指针代替普通指针的二叉树**



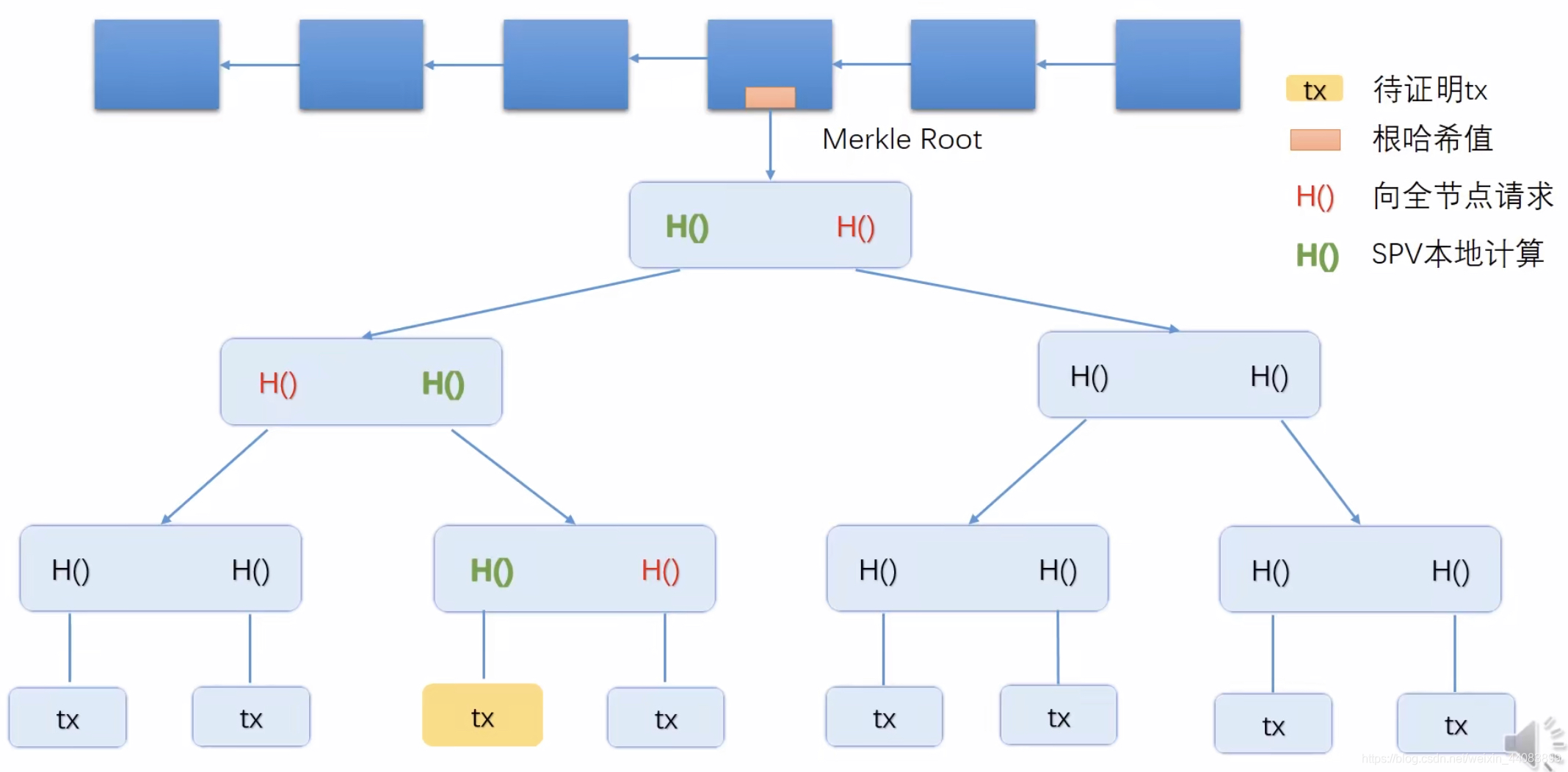
Merkle Tree和普通二叉树的区别就是用哈希指针代替普通指针，最底下的一层叶子结点是数据块，其上的若干层非叶子结点都是存储哈希指针的。一个非叶子结点中的两个哈希值拼在一起，再取一个哈希值，也就是指向该结点的父结点存储的哈希指针的哈希值。对根节点也可以取一个哈希值，称为根哈希值（root hash）。

和区块链一样，在Merkle Tree中，只要记录下根哈希值，就能检测出对树中任何部位的修改，也就是用根哈希值保护了整棵树上没有篡改。

1. **Merkle tree的作用**

**在比特币系统中，Merkle tree主要用于进行Merkle proof。**

在区块链中，每个区块分为两部分，块头（block header）和块身（block body）。在块头中存储了这个区块所包含的所有交易组成的Merkle Tree的根哈希值。只在块身中存储了交易列表。



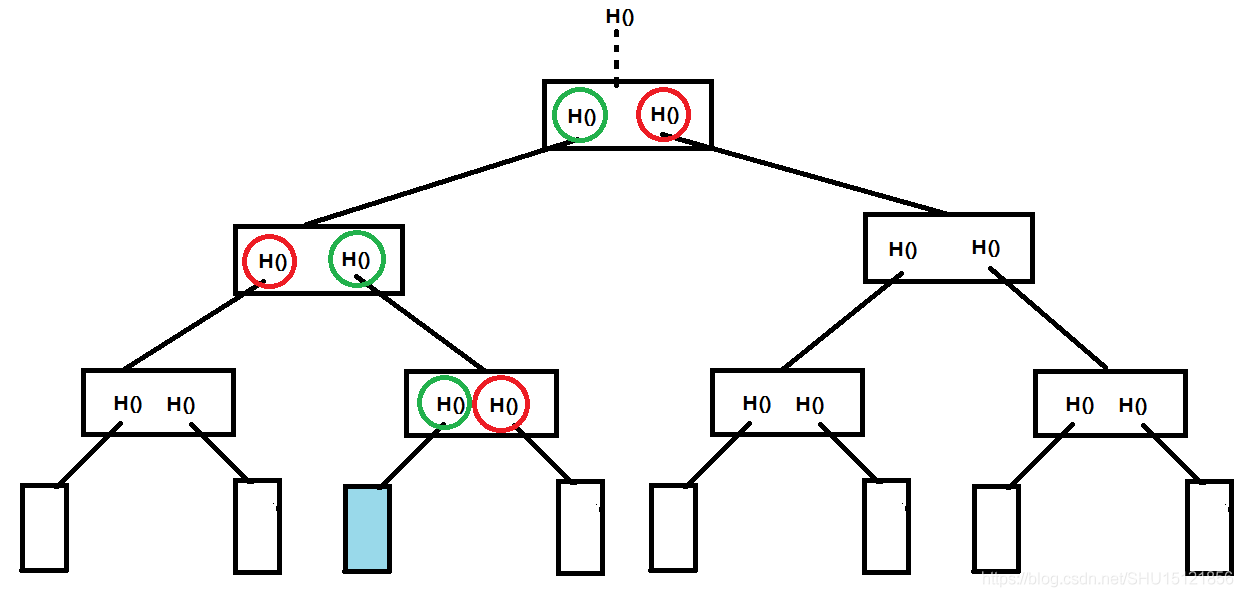
在比特币的区块链中，**区块可以分为两大类：全结点和轻结点**。全结点是既有块头又有块身的区块，保存了交易的具体信息；而轻结点只保存了块头，没有保存块身。例如手机上的比特币钱包就使用的是轻结点。

1. **Merkle proof的实现**

如何向轻结点证明某个交易是写入了区块链的？例如有人向自己转账比特币，在手机上的比特币钱包使用的是轻结点，**轻结点没有块身也就没有存储交易列表，只有一个根哈希值**，如何证明这笔交易是真实存在某个区块中的？

Merkle proof要做的就是**通过区块中的块头中的Merkle Tree的根哈希值证明某个交易是存在于这个区块所对应的Merkle Tree中的**，简单的说就是验证一下Merkle Tree中存在某个交易，这也叫作proof of membership或proof of inclusion。这涉及在Merkle Tree中从指定交易的数据结点到根节点的路径。

在下图中，天蓝色数据结点是待证明存在的交易，这里轻结点只有一个，要获取根哈希值是不够的，因此**轻结点需要向全结点请求下图中标红的三个哈希值**，然后就只需在本地为交易的数据结点向上一步步计算和拼接计算哈希值**(三处绿色的哈希是通过计算的得到的**)，最终和根哈希值对比，来知晓这个交易是不是真实存在这个Merkle Tree中的了。



Merkle proof **验证Merkle Tree中某个交易存在**，假设有N个交易（叶子结点），Merkle proof的时间复杂度是。

假如要在Merkle proof中搜索某个交易，即[**证明其不存在，或在不知道它的路径上的哈希值的情况下证明其存在**]，最暴力的就是验证和搜索整个Merkle Tree，时间复杂度是 。如果在区块保存交易记录时没有任何使用任何特殊方法是，我们也只有这一种方法，但是如何区块在保存交易记录时是按照哈希值由大到小的方式进行保存(**sorted Merkle Tree** )的话，我们就可以采用二分查找的方法简化验证过程：

首先我们需要计算出要搜索的记录的哈希值，然后通过二分查找(也可以是其他查找方法)锁定该哈希值在所有叶子节点的范围，找到离他最近的两个交易的哈希值(一左一右，假如是H1和H2)，此时我们采用与Merkle proof中相同的方法，只不过需要同时验证H1和H2，计算H1和H2上层双亲节点哈希，……直到根节点哈希，如果计算得到的根哈希与区块中根哈希一致，说明H1和H2确实是紧挨着的，就可以说明要搜索的这个交易并不存在与区块中。

比特币中的Merkle Tree不是sorted Merkle Tree，因为没有上面说的这种需求（证明其不存在，或在不知道它的路径上的哈希值的情况下证明其存在）。