# 9.1简介

区块链的数据结构是由包含交易信息的区块按照从远及近的顺序有序链接起来的。它可以被存储为平面文件（flat file），或是存储在一个简单数据库中。比特币核心客户端使用Google的LevelDB数据库存储区块链元数据。区块被从远及近有序地链接在这个链条里，每个区块都指向前一个区块。区块链经常被视为一个垂直的栈，第一个区块作为栈底的首区块，随后每个区块都被放置在之前的区块之上。用栈来形象化表示区块依次堆叠这一概念后，我们便可以使用 一些术语，例如：“高度”来表示区块与首区块之间的距离；以及“顶部”或“顶端”来表示最新添加的区块。

对每个区块头进行SHA256加密哈希，可生成一个哈希值。通过这个哈希值，可以识别出区块链中的对应区块。同时， 每一个区块都可以通过其区块头的“父区块哈希值”字段引用前一区块（父区块）。也就是说，每个区块头都包含它的父区块哈希值。这样把每个区块链接到各自父区块的哈希值序列就创建了一条一直可以追溯到第一个区块（创世区块）的链条。

虽然每个区块只有一个父区块，但可以暂时拥有多个子区块。每个子区块都将同一区块作为其父区块，并且在“父区块哈希值”字段中具有相同的（父区块）哈希值。一个区块出现多个子区块的情况被称为“区块链分叉”。区块链分叉只是暂时状态，只有当多个不同区块几乎同时被不同的矿工发现时才会发生（参见“区块链分叉”）。最终，只有一个子区块会成为区块链的一部分，同时解决了“区块链分叉”的问题。尽管一个区块可能会有不止一个子区块，但每一区块只有 一个父区块，这是因为一个区块只有一个“父区块哈希值”字段可以指向它的唯一父区块。

由于区块头里面包含“父区块哈希值”字段，所以当前区块的哈希值也受到该字段的影响。如果父区块的身份标识发生变化，子区块的身份标识也会跟着变化。当父区块有任何改动时，父区块的哈希值也发生变化。这将迫使子区块的“父区块哈希值”字段发生改变，从而又将导致子区块的哈希值发生改变。而子区块的哈希值发生改变又将迫使孙区块的“父区块哈希值”字段发生改变，又因此改变了孙区块哈希值，以此类推。一旦一个区块有很多代以后，这种瀑布效应将保证该区块不会被改变，除非强制重新计算该区块所有后续的区块。正是这样的重新计算需要耗费巨大的计算量，所以一个长区块链的存在可以让区块链的历史不可改变，这也是比特币安全性的一个关键特征。

你可以把区块链想象成地质构造中的地质层或者是冰川岩芯样品。表层可能会随着季节而变化，甚至在沉积之前就被风吹走了。但是越往深处，地质层就变得越稳定。到了几百英尺深的地方，你看到的将是保存了数百万年但依然保持历史原状的岩层。在区块链里，最近的几个区块可能会由于区块链分叉所引发的重新计算而被修改。最新的六个区块就像几英寸深的表土层。但是，超过这六块后，区块在区块链中的位置越深，被改变的可能性就越小。在100个区块以后，区块链已经足够稳定，这时Coinbase交易（包含新挖出的比特币的交易）可以被支付。几千个区块（一个月）后的区块链将变成确定的历史，永远不会改变。

# 9.2 区块结构

区块是一种被包含在公开账簿（区块链）里的聚合了交易信息的容器数据结构。它由一个包含元数据的区块头和紧跟其后的构成区块主体的一长串交易列表组成。区块头是80字节，而平均每个交易至少是250字节，而且平均每个区块至少包含超过500个交易。因此，一个包含所有交易的完整区块比区块头大1000倍。表7-1描述了一个区块结构。

# 9.3 区块头

区块头由三组区块元数据组成。首先是一组引用父区块哈希值的数据，这组元数据用于将该区块与区块链中前一区块相连接。第二组元数据，即难度、时间戳和nonce，与挖矿竞争相关，详见挖矿章节。第三组元数据是merkle树根（一种用来有效地总结区块中所有交易的数据结构）。表7-2描述了区块头的数据结构。

Nonce、难度目标和时间戳会用于挖矿过程，更多细节将在挖矿章节讨论。

## 9.4 区块标识符：区块头哈希值和区块高度

区块主标识符是它的加密哈希值，一个通过SHA256算法对区块头进行二次哈希计算而得到的数字指纹。产生的32字节哈希值被称为区块哈希值，但是更准确的名称是：区块头哈希值，因为只有区块头被用于计算。例如:000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f是第一个比特币区块的区块哈希值。区块哈希值可以唯一、明确地标识一个区块，并且任何节点通过简单地对区块头进行哈希计算都可以独立地获取该区块哈希值。

请注意，区块哈希值实际上并不包含在区块的数据结构里，不管是该区块在网络上传输时，抑或是它作为区块链的一部分被存储在某节点的永久性存储设备上时。相反，区块哈希值是当该区块从网络被接收时由每个节点计算出来的。区块的哈希值可能会作为区块元数据的一部分被存储在一个独立的数据库表中，以便于索引和更快地从磁盘检索区块。

第二种识别区块的方式是通过该区块在区块链中的位置，即“区块高度（block height）”。第一个区块，其区块高度为 0，和之前哈希值000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f所引用的区块为同一 个区块。因此，区块可以通过两种方式被识别：区块哈希值或者区块高度。每一个随后被存储在第一个区块之上的区块在区块链中都比前一区块“高”出一个位置，就像箱子一样,一个接一个堆叠在其他箱子之上。2017年1月1日的区块高度大约是 446,000，说明已经有446,000个区块被堆叠在2009年1月创建的第一个区块之上。

和区块哈希值不同的是，区块高度并不是唯一的标识符。虽然一个单一的区块总是会有一个明确的、固定的区块高度， 但反过来却并不成立，一个区块高度并不总是识别一个单一的区块。两个或两个以上的区块可能有相同的区块高度，在区块链里争夺同一位置。这种情况在“区块链分叉”一节中有详细讨论。区块高度也不是区块数据结构的一部分， 它并不被存储在区块里。当节点接收来自比特币网络的区块时，会动态地识别该区块在区块链里的位置（区块高度）。 区块高度也可作为元数据存储在一个索引数据库表中以便快速检索。

**提示**一个区块的区块哈希值总是能唯一地识别出一个特定区块。一个区块也总是有特定的区块高度。但是，一 个特定的区块高度并不一定总是能唯一地识别出一个特定区块。更确切地说，两个或者更多数量的区块也许会为了区块链中的一个位置而竞争。

## 9.5 创世区块

区块链里的第一个区块创建于2009年，被称为创世区块。它是区块链里面所有区块的共同祖先，这意味着你从任一区块，循链向后回溯，最终都将到达创世区块。

因为创世区块被编入到比特币客户端软件里，所以每一个节点都始于至少包含一个区块的区块链，这能确保创世区块不会被改变。每一个节点都“知道”创世区块的哈希值、结构、被创建的时间和里面的一个交易。因此，每个节点都把该区块作为区块链的首区块，从而构建了一个安全的、可信的区块链。

在*chainparams.cpp*里可以看到创世区块被编入到比特币核心客户端里。

创世区块的哈希值为： 0000000000 19d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

你可以在任何区块浏览网站搜索这个区块哈希值，如blockchain.info，你会发现一个描述这一区块内容的页面，该页面的链接包含了这个区块哈希值：

https://blockchain.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

https://blockexplorer.com/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

在命令行使用比特币核心客户端:

```
$ bitcoindgetblock 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
{
    "hash":"000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
    "confirmations":308321,
    "size":285,
    "height":0,
    "version":1,
    "merkleroot":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
    "tx":["4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"],
    "time":1231006505,
    "nonce":2083236893,
    "bits":"1d00ffff",
    "difficulty":1.00000000,
    "nextblockhash":"00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048"
}
```

创世区块包含一个隐藏的信息。在其Coinbase交易的输入中包含这样一句话“The Times 03/Jan/2009 Chancellor on brink of second bailout forbanks.”这句话是泰晤士报当天的头版文章标题，引用这句话，既是对该区块产生时间的说明，也可视为半开玩笑地提醒人们一个独立的货币制度的重要性，同时告诉人们随着比特币的发展，一场前所未有的世界性货币革命将要发生。该消息是由比特币的创立者中本聪嵌入创世区块中。

# 9.6 区块链接成为区块链

比特币的全节点在本地保存了区块链从创世区块起的完整副本。随着新的区块的产生，该区块链的本地副本会不断地更新用于扩展这个链条。当一个节点从网络接收传入的区块时，它会验证这些区块，然后链接到现有的区块链上。为建立一个连接，一个节点将检查传入的区块头并寻找该区块的“父区块哈希值”。

让我们假设，例如，一个节点在区块链的本地副本中有277,314个区块。该节点知道最后一个区块为第277,314个区 块，这个区块的区块头哈希值为: 00000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249。 然后该比特币节点从网络上接收到一个新的区块，该区块描述如下:

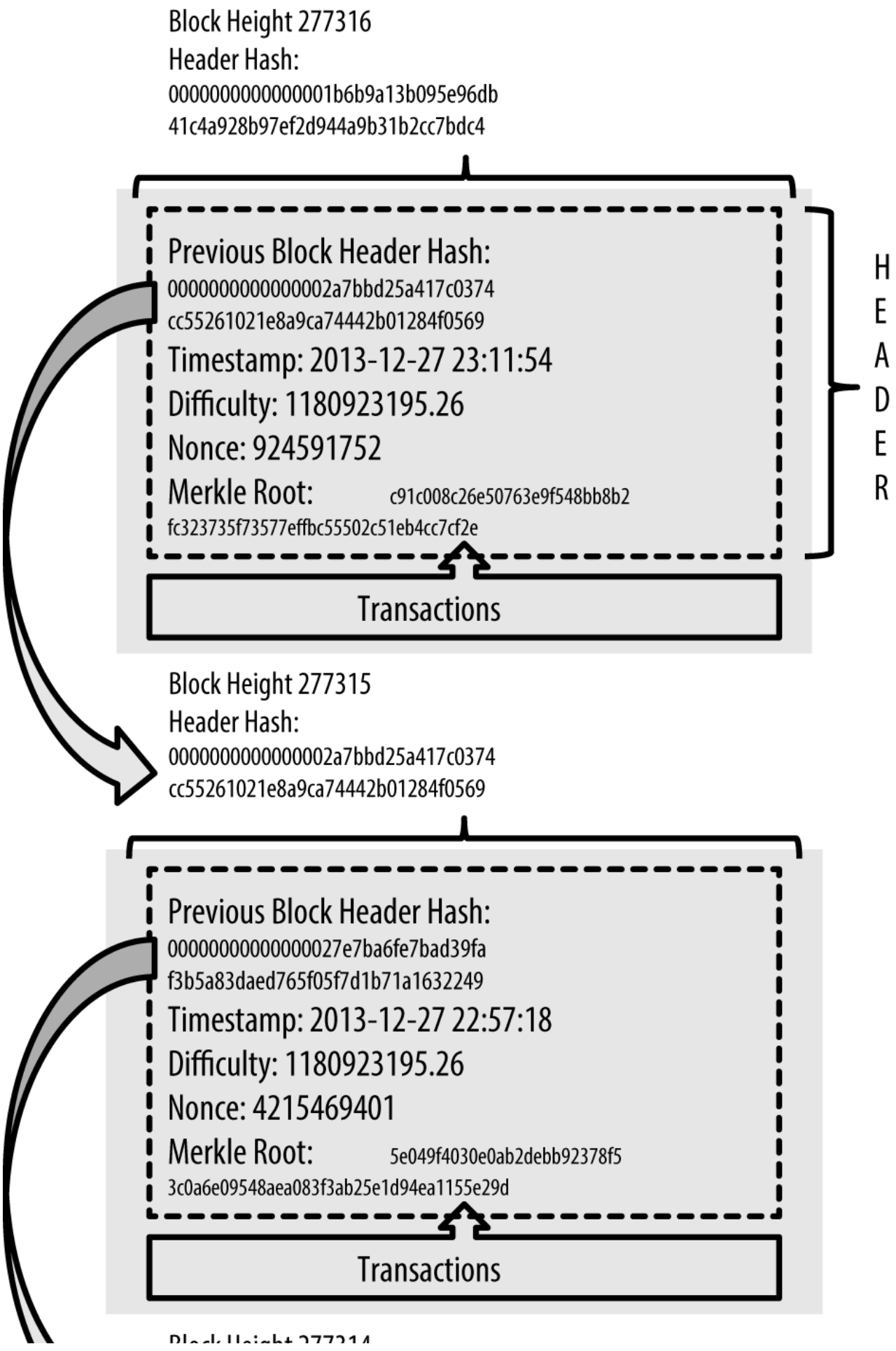
```
{
"size" : 43560,
"version" : 2,
"previousblockhash" :
    "00000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249",
"merkleroot" :
    "5e049f4030e0ab2debb92378f53c0a6e09548aea083f3ab25e1d94ea1155e29d",
"time" : 1388185038,
"difficulty" : 1180923195.25802612,
"nonce" : 4215469401,
"tx" : [
    "257e7497fb8bc68421eb2c7b699dbab234831600e7352f0d9e6522c7cf3f6c77",

 [... many more transactions omitted ...]

    "05cfd38f6ae6aa83674cc99e4d75a1458c165b7ab84725eda41d018a09176634"
]
```

```
}
```

对于这一新的区块，节点会在“父区块哈希值”字段里找出包含它的父区块的哈希值。这是节点已知的哈希值，也就是第 277314块区块的哈希值。故这个新区块是这个链条里的最后一个区块的子区块，因此现有的区块链得以扩展。节点将新的区块添加至链条的尾端，使区块链变长到一个新的高度277,315。图9-1显示了通过“父区块哈希值”字段进行连接三个区块的链。


Block Height 277316
Header Hash:
0000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4
Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e
Transactions
HEADER
Block Height 277315
Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Previous Block Header Hash:
00000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce: 4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d
Transactions

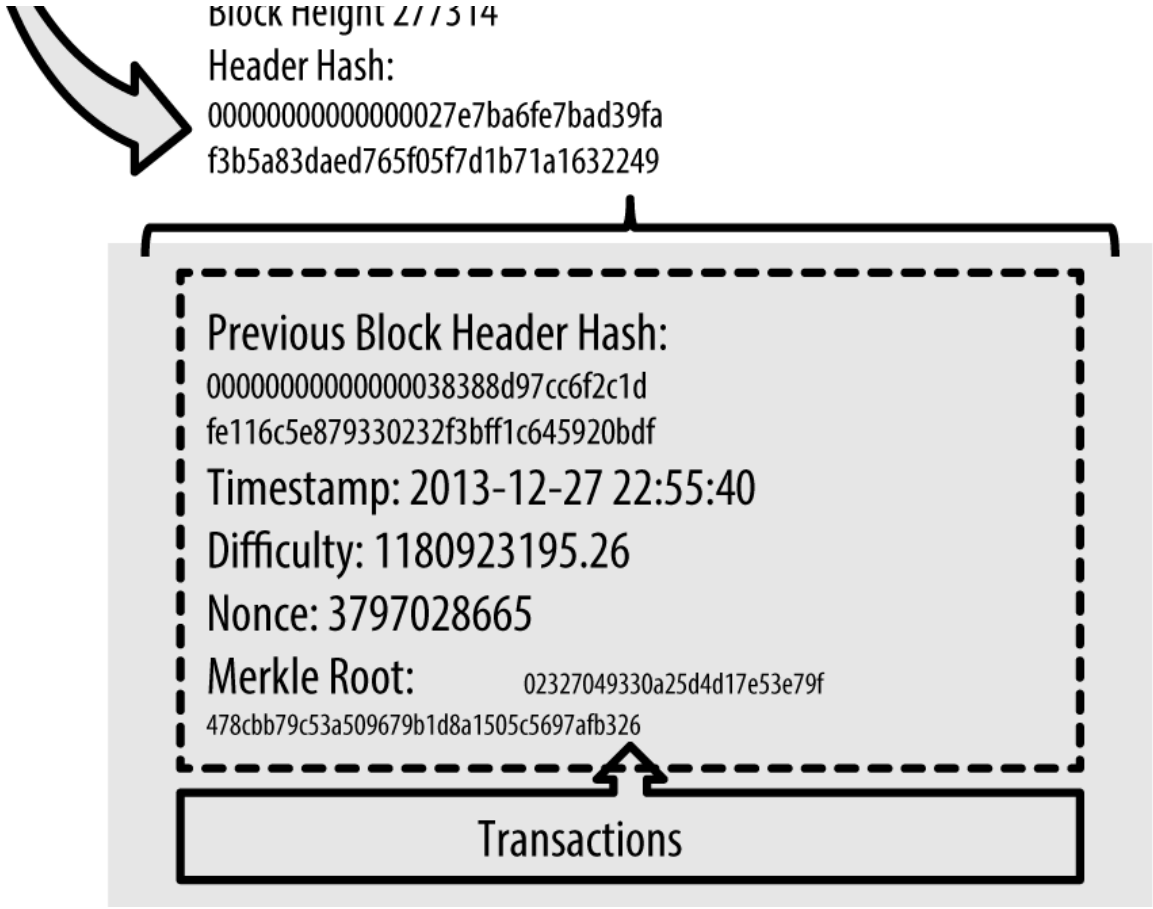



图9-1通过引用前面的区块头哈希连接成区块链

# 9.7 Merkle 树

区块链中的每个区块都包含了产生于该区块的所有交易，且以Merkle树表示。

Merkle树是一种哈希二叉树，它是一种用作快速归纳和校验大规模数据完整性的数据结构。这种二叉树包含加密哈希值。术语“树”在计算机学科中常被用来描述一种具有分支的数据结构，但是树常常被倒置显示，“根”在图的上部同时“叶子”在图的下部，你会在后续章节中看到相应的例子。

在比特币网络中，Merkle树被用来归纳一个区块中的所有交易，同时生成整个交易集合的数字指纹，且提供了一种校验区块是否存在某交易的高效途径。生成一棵完整的Merkle树需要递归地对哈希节点对进行哈希，并将新生成的哈希节点插入到Merkle树中，直到只剩一个哈希节点，该节点就是Merkle树的根。在比特币的Merkle树中两次使用到了SHA256 算法，因此其加密哈希算法也被称为double-SHA256。

当N个数据元素经过加密后插入Merkle树时，你至多计算2*log~2~(N) 次就能检查出任意某数据元素是否在该树中，这使得该数据结构非常高效。

Merkle树是自底向上构建的。在如下的例子中，我们从A、B、C、D四个构成Merkle树树叶的交易开始，如图9-2。

图9-2计算默克树中的节点

所有的交易都并不存储在Merkle树中，而是将数据哈希化，然后将哈希值存储至相应的叶子节点。这些叶子节点分别是H~A~、H~B~、H~C~和H~D~：

HA = SHA256(SHA256(Transaction A))

将相邻两个叶子节点的哈希值串联在一起进行哈希，这对叶子节点随后被归纳为父节点。 例如，为了创建父节点H~AB~，子节点 A和子节点B的两个32字节的哈希值将被串联成64字节的字符串。随后将字符串进行两次哈希来产生父节点的哈希值:

HAB = SHA256(SHA256(H~A~ + H~B~))

继续类似的操作直到只剩下顶部的一个节点，即Merkle根。产生的32字节哈希值存储在区块头，同时归纳了四个交易的所有数据。图9-2展示了如何通过成对节点的哈希值计算Merkle树的根。

因为Merkle树是二叉树，所以它需要偶数个叶子节点。如果仅有奇数个交易需要归纳，那最后的交易就会被复制一份以构成偶数个叶子节点，这种偶数个叶子节点的树也被称为平衡树。如图9-3所示，C节点被复制了一份。

图9-3复制一个数据元素可以实现偶数个数据元素

由四个交易构造Merkle树的方法同样适用于从任意交易数量构造Merkle树。在比特币中，在单个区块中有成百上千的交易是非常普遍的，这些交易都会采用同样的方法归纳起来，产生一个仅仅32字节的数据作为Merkle根。在图9-4中，你会看见一个从16个交易形成的树。需要注意的是，尽管图中的根看起来比所有叶子节点都大，但实际上它们都是32字节的相同大小。无论区块中有一个交易或者有十万个交易，Merkle根总会把所有交易归纳为32字节。

图9-4Merkle树汇总了许多数据元素

为了证明区块中存在某个特定的交易，一个节点只需要计算log~2~(N)个32字节的哈希值，形成一条从特定交易到树根的认证路径或者Merkle路径即可。随着交易数量的急剧增加，这样的计算量就显得异常重要，因为相对于交易数量的增长，以基底为2的交易数量的对数的增长会缓慢许多。这使得比特币节点能够高效地产生一条10或者12个哈希值（320~384字节）的路径，来证明了在一个巨量字节大小的区块中上千交易中的某笔交易的存在。

在图9-5中，一个节点能够通过生成一条仅有4个32字节哈希值长度（总128字节）的Merkle路径，来证明区块中存在一 笔交易K。该路径有4个哈希值（在图9-5中由蓝色标注）H~L~、H~IJ~、H~MNOP~和H~ABCDEFGH~。由这4个哈希值产生的认证路径，再通过计算另外四对哈希值H~KL~、H~IJKL~、H~IJKLMNOP~和Merkle树根（在图中由虚线标注），任何节点都能证明H~K~（在图中由绿色标注）包含在Merkle根中。

图9-5用于证明包含数据元素的merkle路径

例9-1中的代码借用libbitcoin库中的一些辅助程序，演示了从叶子节点哈希至根创建整个Merkle树的过程。

例9-1 构造Merkle树

link:code/merkle.cpp[]

例9-2展示了编译以及运行上述代码后的结果

```
\ # Compile the merkle.cpp code
$ g++ -o merkle merkle.cpp (pkg-config --cflags --libs libbitcoin)
\ # Run the merkle executable
$ ./merkle
Current merkle hash list:
 32650049a0418e4380db0af81788635d8b65424d397170b8499cdc28c4d27006
 30861db96905c8dc8b99398ca1cd5bd5b84ac3264a4e1b3e65afa1bcee7540c4
Current merkle hash list:
 d47780c084bad3830bcdaf6eace035e4c6cbf646d103795d22104fb105014ba3
Result: d47780c084bad3830bcdaf6eace035e4c6cbf646d103795d22104fb105014ba3
```

Merkle树的高效随着交易规模的增加而变得异常明显。表9-3展示了为了证明区块中存在某交易而所需转化为Merkle路径的数据量。

从表中可以看出，当区块大小由16笔交易（4KB）急剧增加至65,535笔交易（16MB）时，为证明交易存在的Merkle路径长度增长极其缓慢，仅仅从128字节到512字节。有了Merkle树，一个节点能够仅下载区块头（80字节/区块），然后通过从一个满节点回溯一条小的Merkle路径就能认证一笔交易的存在，而不需要存储或者传输大量区块链中大多数内容，这些内容可能有几个G的大小。这种不需要维护一条完整的区块链的节点，又被称作简单支付验证（SPV）节点，它不需要下载整个区块而通过Merkle路径去验证交易的存在。

# 9.8 Merkle树和简单支付验证（SPV）

Merkle树被SPV节点广泛使用。SPV节点不保存所有交易也不会下载整个区块，仅仅保存区块头。它们使用认证路径或者Merkle路径来验证交易存在于区块中，而不必下载区块中所有交易。

例如，一个SPV节点想知道它钱包中某个比特币地址即将到达的支付。该节点会在节点间的通信链接上建立起bloom过滤器，限制只接受含有目标比特币地址的交易。当节点探测到某交易符合bloom过滤器，它将以Merkleblock消息的形式发送该区块。Merkleblock消息包含区块头和一条连接目标交易与Merkle根的Merkle路径。SPV节点能够使用该路径找到与该交易相关的区块，进而验证对应区块中该交易的有无。SPV节点同时也使用区块头去关联区块和区块链中的其余区块。这两种关联，交易与区块、区块和区块链，就可以证明交易存在于区块链。简而言之，SPV节点会收到少于1KB的有关区块头和Merkle路径的数据，其数据量比一个完整的区块（目前大约有1MB）少了一千多倍。

# 9.9比特币的测试区块链

你可能会惊讶地发现，有多个比特币区块链。 2009年1月3日由Satoshi Nakamoto创建的“主要”比特币块链，即本章研究的创世区块所在的网络，被称为主干网。另外还有其他用于测试的比特币区块链：现存的有testnet，segnet和regtest。我们依次看看每一个。

## 9.9.1Testnet——比特币的试验场

Testnet是用于测试的区块链，网络和货币的总称。 testnet是一个功能齐全的在线P2P网络，包括钱包，测试比特币（testnet币），挖矿以及类似主干网的所有其他功能。实际上它和主网只有两个区别：testnet币是毫无价值的，挖掘难度足够低，任何人都可以相对容易地使用testnet币）。

任何打算在比特币主干网上用于生产的软件开发都应该首先在testnet上用测试币进行测试。这样可以保护开发人员免受由于软件错误而导致的金钱损失，也可以保护网络免受由于软件错误导致的意外攻击。

然而，保持测试币的无价值和易挖掘并不容易。尽管有来自开发商的呼吁，但还是有人使用先进的设备（GPU和ASIC）在testnet上挖矿。这就增加了难度，使用CPU挖矿不可能，导致获取测试币非常困难，以致于人们开始赋予其一定价值，所以测试币并不是毫无价值。结果，时不时地testnet必须被报废并重新从创始区块启动，重新进行难度设置。

目前的testnet被称为testnet3，是testnet的第三次迭代，于2011年2月重启，重置了之前的testnet网络的难度。

请记住，testnet3是一个大区块链，在2017年初超过20 GB。完全同步需要一天左右的时间，并占用您的计算机资源。它不像主干网，也不是“轻量级”。运行testnet节点的一个好方法就是将其运行为一个专用的虚拟机镜像（例如，VirtualBox，Docker，Cloud Server等）。

### 9.9.1.1使用testnet

像几乎所有其他比特币软件一样，Bitcoin Core完全支持在testnet网络运行而不是只能在主干网上运行，还允许您进行测试币挖矿并运行一个testnet全节点。

如果要在testnet上启动Bitcoin Core，而不是在主干网启动，您可以使用testnet开关：

$ bitcoind -testnet

在日志中，您应该会看到，bitcoind正在默认bitcoind目录的testnet3子目录中构建一个新的区块链：

bitcoind: Using data directory /home/username/.bitcoin/testnet3

要连接bitcoind，可以使用bitcoin-cli命令行工具，但是要记得切换到testnet模式：

```
$ bitcoin-cli -testnet getinfo

{
  "version": 130200,
  "protocolversion": 70015,
  "walletversion": 130000,
  "balance": 0.00000000,
  "blocks": 416,
  "timeoffset": 0,
  "connections": 3,
  "proxy": "",
  "difficulty": 1,
  "testnet": true,
  "keypoololdest": 1484801486,
  "keypoolsize": 100,
  "paytxfee": 0.00000000,
  "relayfee": 0.00001000,
  "errors": ""
}
```

您还可以使用getblockchaininfo命令确认testnet3区块链的详细信息和同步进度：

```
$ bitcoin-cli -testnet getblockchaininfo

{
  "chain": "test",
  "blocks": 1088,
```

```
  "headers": 139999,
  "bestblockhash": "0000000063d29909d475a1c4ba26da64b368e56cce5d925097bf3a2084370128",
  "difficulty": 1,
  "mediantime": 1337966158,
  "verificationprogress": 0.001644065914099759,
  "chainwork": "0000000000000000000000000000000000000000000000000044104410441",
  "pruned": false,
  "softforks": [

  [...]
```

在testnet3上，你也可以运行使用其他语言和框架实现的全节点来实验和学习，例如btcd（用Go编写）和bcoin（用JavaScript编写）。

在2017年初，testnet3支持主网的所有功能，也包括在主干网络上尚未激活的隔离见证（Segregated Witness（见[segwit]隔离见证章节））。 因此，testnet3也可用于测试隔离见证功能。

## 9.9.2 Segnet—隔离见证测试网络

2016年，启动了一个特殊用途的测试网络，以帮助开发和测试隔离见证（也称为segwit;见[segwit]）。 该测试区块链称为segnet，可以通过运行Bitcoin Core的特殊版本（分支）来连接。

由于已经将segwit添加到testnet3中，因此后来不再使用segnet来测试segwit功能。

在将来，我们可能会看到其他专门用于测试单个功能或主要架构更改（如segnet）的测试网络区块链。

## 9.9.3 Regtest--本地区块链

Regtest代表“回归测试”，是一种比特币核心功能，允许您创建本地区块链以进行测试。 与testnet3（它是一个公共和共享的测试区块链）不同，regtest区块链旨在作为本地测试的封闭系统运行。 您从头开始启动regtest区块链，创建一个本地的创世区块。 您可以将其他节点添加到网络中，或者使用单个节点运行它来测试Bitcoin Core软件。

要在regtest模式下启动Bitcoin Core，您可以使用regtest标志：

$ bitcoind -regtest

就像使用testnet一样，Bitcoin Core将在bitcoind默认目录的regtest子目录下初始化一个新的区块链：

bitcoind: Using data directory /home/username/.bitcoin/regtest

要使用命令行工具，还需要指定regtest标志。 我们来试试getblockchaininfo命令来检查regtest区块链：

```
$ bitcoin-cli -regtest getblockchaininfo

{
  "chain": "regtest",
  "blocks": 0,
  "headers": 0,
  "bestblockhash": "0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206",
  "difficulty": 4.656542373906925e-10,
  "mediantime": 1296688602,
  "verificationprogress": 1,
  "chainwork": "0000000000000000000000000000000000000000000000000000000000000002",
  "pruned": false,
  [...]
```

你可以看到，还没有任何区块。 让我们开始挖一些（500块），赚取奖励：

```
$ bitcoin-cli -regtest generate 500

[
  "7afed70259f22c2bf11e406cb12ed5c0657b6e16a6477a9f8b28e2046b5ba1ca",
  "1aca2f154a80a9863a9aac4c72047a6d3f385c4eec5441a4aafa6acaa1dada14",
  "4334ecf6fb022f30fbd764c3ee778fabbd53b4a4d1950eae8a91f1f5158ed2d1",
  "5f951d34065efeaf64e54e91d00b260294fcdfc7f05dbb5599aec84b957a7766",
  "43744b5e77c1dfece9d05ab5f0e6796ebe627303163547e69e27f55d0f2b9353",
   [...]
  "6c31585a48d4fc2b3fd25521f4515b18aefb59d0def82bd9c2185c4ecb754327"
]
```

挖掘所有这些块只需要几秒钟，这样就可以很容易地进行测试。 如果您检查您的钱包余额，您将看到您获得了前400个区块的奖励（Coinbase的奖励必须挖满100块之后才能花费）：

```
$ bitcoin-cli -regtest getbalance

12462.50000000
```

# 9.10使用测试块链进行开发

Bitcoin的各种区块链（regtest，segnet，testnet3，以及主干网）为比特币开发提供了一系列测试环境。 无论您是开发比特币核心还是另一个全节点共识客户端，诸如钱包，交易所，电子商务网站等应用程序，甚至开发新颖的智能合同和复杂的脚本等等，请使用测试区块链网进行开发。

您可以使用测试区块链来建立开发管道。 在开发它时，建议在本地测试代码。

一旦您准备好在公共网络上尝试，请切换到testnet，将您的代码暴露在更加动态的环境中，并提供更多样化的代码和应用程序。

最后，一旦您确信您的代码正常工作，请切换到主网以实现生产部署。

当您进行变更，改进，错误修复等操作时，再次启动这个开发管道，首先在regtest上部署每个变更，然后在testnet上进行测试，最后实现生产。