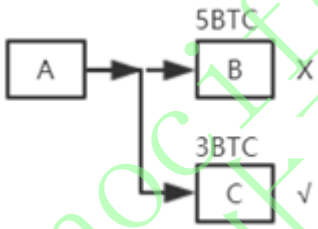
北京大学肖臻老师《区块链技术与应用》公开课笔记

> 比特币具体实现篇，对应肖老师视频：click here 全系列笔记请见：click here **About Me:**点击进入我的Personal Page

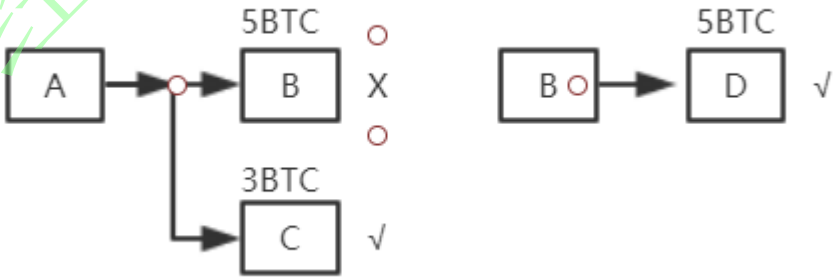
区块链是一个去中心化的账本，比特币采用了 **基于交易的账本模式** 。然而，系统中并无显示记录账户包含比特币数，实际上其需要通过交易记录进行推算。在比特币系统中，全节点需要维护一个名为 **UTXO(Unspent Transaction Output尚未被花掉的交易输出)** 的数据结构。

> 如图，A转给B五个BTC，转给C3个BTC，B将5个BTC花掉，则该交易记录不保存在UTXO中，C没有花掉，则该交易记录保存在UTXO中
>
> 
>

UTXO集合中每个元素要给出产生这个输出的交易的哈希值，以及其在交易中是第几个输出。通过这两个信息，便可以定位到UTXO中的输出。

> 为什么要维护这样一个数据结构？？？为了防范“双花攻击”，判断一个交易是否合法，要查一下想要花掉的BTC是否在该集合中，只有在集合中才是合法的。如果想要花掉的BTC不在UTXO中，那么说明这个BTC要么根本不存在，要么已经被花过。所以，全节点需要在内存中维护一个UTXO，从而便于快速检测double spending（双花攻击）。

每个交易会消耗输出，但也会产生新的输出。

> 如图，A转给B5个BTC，之后B将其转给D，则UTXO中会删掉A->B这一交易记录，同时会添加B->D这一交易记录。
>
> 
>
>
> 假如有人收到BTC转账，但一直不花，那么这个信息会一直保存在UTXO中。这种情况可能是该用户不想花这些BTC(如：中本聪)，也有可能是忘记了私钥导致无法花掉。所以，UTXO是逐渐增大的，但该数据目前来说，一个普通的服务器硬盘中是可以完全保存这些数据的。

每个交易可以有多个输入，也可以有多个输出，但输入之和要等于输出之和（total inputs = total outputs）。 存在一些交易的total inputs 略大于 total outputs，这部分差额便作为交易费，给了获得记账权的节点。在公开课笔记4中最后提及到“区块中保存交易记录，如果仅仅设置出块奖励，那么，会不会存在节点只想发布区块获得出块奖励而不想打包交易？”因此，BTC系统设计了Tranction fee（交易费），对于获得记账权节点来说，除了出块奖励之外，还可以得到打包交易的交易费。但目前来说，交易费远远小于出块奖励。等到未来出块奖励变少，可能区块链的维护便主要依赖于交易费了。

> BTC系统中每21万个区块，BTC出块奖励减半。根据下图计算，基本上出块奖励每4年减半。
>
> 
>

> 比特币是基于交易的模式，与之对应，还有一种基于账户的模式（如：以太坊）。基于账户的模式要求，系统中显示记录账户余额。也就是说，可以直接查询当前账户余额是多少货币。可以看到，比特币这种模式，隐私性较好，但其也付出一定代价。在进行交易时，因为没有账户这一概念，无法知道账户剩余多少BTC,所以必须说明币的来源（防止双花攻击）。而基于账户的模式，则天然地避免了这种缺陷，转账交易就是对一个（多个）账户余额的数字减和另一个（多个）账户余额的数字加

BTC系统中具体的区块信息

如下图所示，为一个区块的信息（取自视频中截图，源自blockchain.info）

- 什么是挖矿？ 可以看到，区块哈希与前一区块哈希都是以一长串0开头的，挖矿本身就是尝试各种nonce，使得产生的区块哈希值小于等于目标阈值。该目标阈值，表示成16进制，就是前面含有一长串的0

下为block header的代码中实现的数据结构。里面的几个域在公开课笔记4中(比特币区块信息)已经解释过了，这里不再赘述。

Block header

```
13  /** Nodes collect new transactions into a block, hash them into a hash tree,
14   * and scan through nonce values to make the block's hash satisfy proof-of-work
15   * requirements.  When they solve the proof-of-work, they broadcast the block
16   * to everyone and the block is added to the block chain.  The first transaction
17   * in the block is a special one that creates a new coin owned by the creator
18   * of the block.
19   */
20  class CBlockHeader
21  {
22  public:
23      // header
24      int32_t nVersion;
25      uint256 hashPrevBlock;
26      uint256 hashMerkleRoot;
27      uint32_t nTime;
28      uint32_t nBits;
29      uint32_t nNonce;
```

source: bitcoin/src/primitives/block.h

可以看到，nonce是一个32位的无符号整型数据，在挖矿时候是通过不断调整nonce进行的，但可以看到，nonce的取值最多为2^{32}种。但并非将这些nonce全部遍历一遍，就一定能找到符合要求的nonce。由于近年来，挖矿人员越来越多，挖矿难度已经调整的比较大了（关于难度调整请关注后续博文，会有专门一篇介绍难度调整），而2^{32}这一搜索空间太小，所以仅调整nonce很大可能找不到正确的结果。

> 还有哪些域可以调整呢?

下图为block header中对各个域的描述。而仅仅调整nonce是不够的，所以这里可以通过修改Merkle Tree的根哈希值来进行调整。

> 思考：打包的交易和顺序确定了，根哈希值不就确定了吗？这个怎么能修改呢?

铸币交易（coinbase交易）

Block headers are serialized in the 80-byte format described below and then hashed as part of Bitcoin's proof-of-work algorithm, making the serialized header format part of the consensus rules.

Bytes	Name	Data Type	Description
4	version	int32_t	The **block version** number indicates which set of block validation rules to follow. See the list of block versions below.
32	previous block header hash	char[32]	A SHA256(SHA256()) hash in internal byte order of the previous block's header. This ensures no previous block can be changed without also changing this block's header.
32	merkle root hash	char[32]	A SHA256(SHA256()) hash in internal byte order. The merkle root is derived from the hashes of all transactions included in this block, ensuring that none of those transactions can be modified without modifying the header. See the merkle trees section below.
4	time	uint32_t	The block time is a Unix epoch time when the miner started hashing the header (according to the miner). Must be strictly greater than the median time of the previous 11 blocks. Full nodes will not accept blocks with headers more than two hours in the future according to their clock.
4	nBits	uint32_t	An encoded version of the target threshold this block's header hash must be less than or equal to. See the nBits format described below.
4	nonce	uint32_t	An arbitrary number miners change to modify the header hash in order to produce a hash less than or equal to the target threshold. If all 32-bit values are tested, the time can be updated or the coinbase transaction can be changed and the merkle root updated.

BTC协议中当前使用版本号(无法修改)

前一区块哈希(无法修改)

Merkle Tree根哈希值

区块产生时间(不要求精确时间，所以有一定调整余地)

挖矿时采用的目标阈值(编码后的版本，只能按照协议要求定期调整)

在公开课笔记4中提及，每个发布区块者可以得到出快奖励，也就是可以在区块中发布一个 **铸币交易(coinbase交易)**,这也是BTC系统中产生新比特币的唯一方式。下为一个铸币交易的内容：

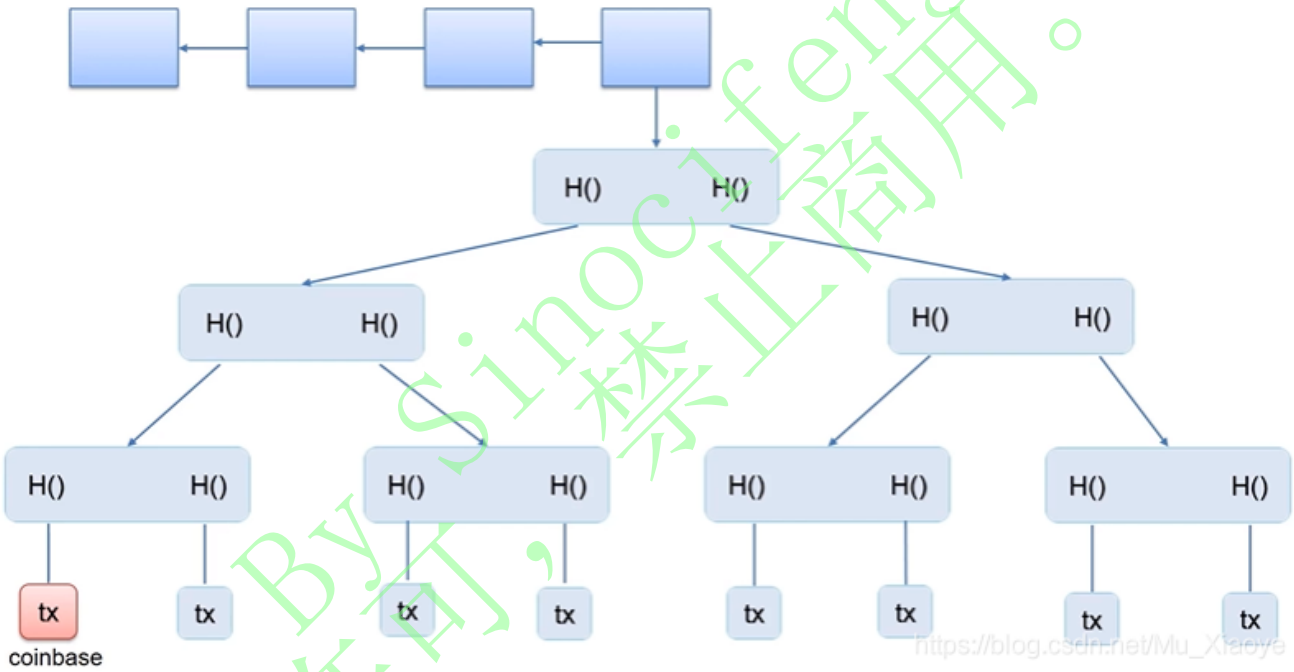
Transaction View information about a bitcoin transaction

Summary	
Size	243 (bytes)
Weight	864
Received Time	2018-06-29 06:17:26
Reward From Block	529709
Scripts	Hide scripts & coinbase
Visualize	View Tree Chart

可以看到，有一个CoinBase域，其

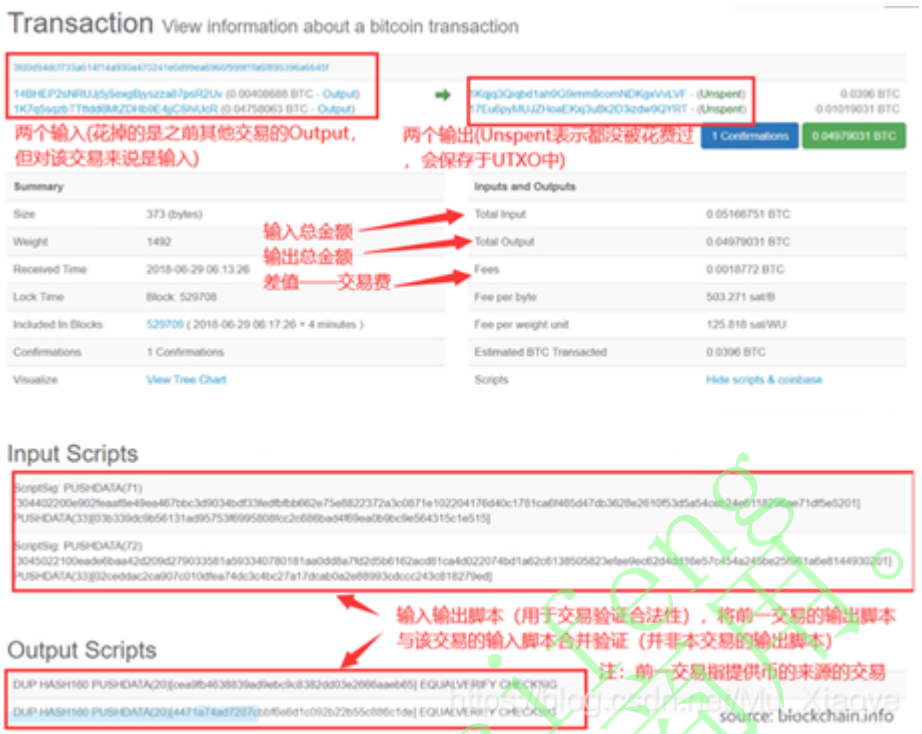
中可以写入任何内容，在这里写什么都没有影响。所以可以在这里添加一些任意信息，便可以实现无法篡改（也无法删除）。（例如：提前写入股票预测结果的哈希值、写入人生感想，写入爱情誓言（无法删除，想想删不掉十年前发表的QQ空间非主流说说是多么痛苦吧，嘿嘿嘿）） 所以，只要我们改变了写入内容，便可以改变Merkle Tree 的根哈希值。

下图为一个小型的区块链，假定左下角交易为coinbase交易，可以看到，该交易发生改变会逐级向上传递，最终导致Merkle Tree根哈希值发生改变。



所以，在实际的挖矿中，包含两层循环。外层循环调整coinbase域（可以规定只将其中前x个字节作为另一个nonce），算出block header中根哈希值后，内层循环再调整nonce。

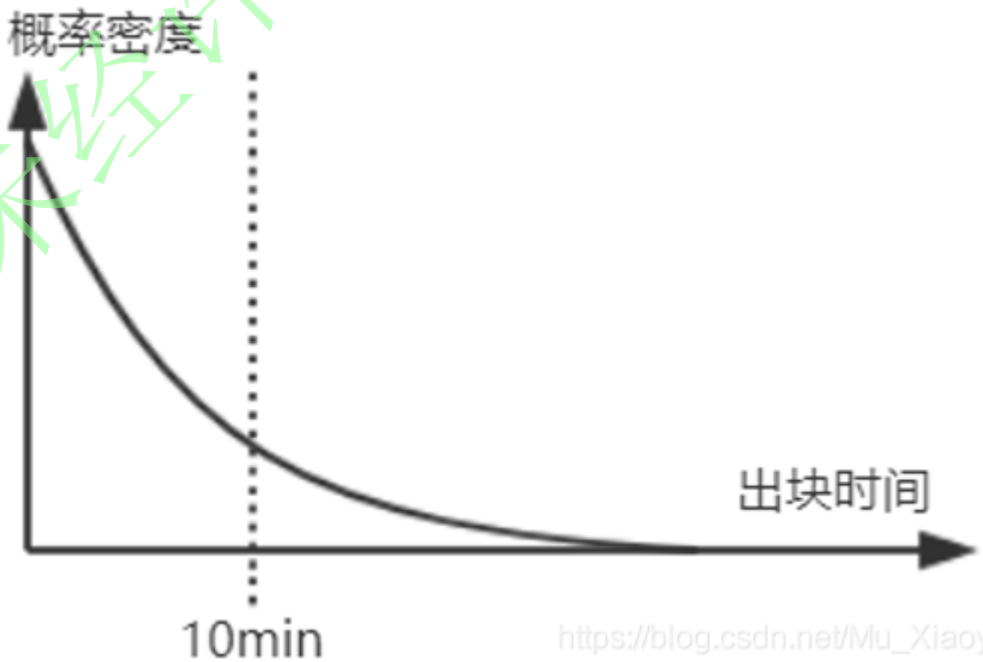
普通转账交易



如果将输入脚本和输出脚本拼接起来可以顺利执行不出现错误，则说明交易合法。

挖矿过程的概率分析

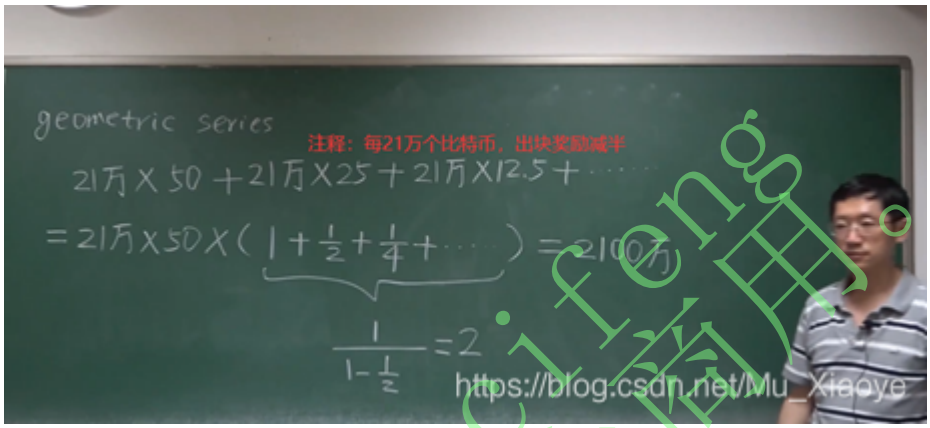
挖矿本质上是不断尝试各种nonce，来求解这样一个puzzle。每次尝试nonce，可以视为一次伯努利试验。最典型的伯努利试验就是投掷硬币，正面和反面朝上概率为p和1-p。在挖矿过程中，一次伯努利试验，成功的概率极小，失败的概率极大。挖矿便是多次进行伯努利试验，且每次随机。这些伯努利试验便构成了a sequence of independent Bernoulli trials(一系列独立的伯努利试验)。根据概率论相关知识知道，伯努利试验本身具有无记忆性。也就是说，无论之前做多少大量试验，对后续继续试验没有任何影响（车牌摇号也是如此，，心痛....）。 对于挖矿来说，便是多次伯努利试验尝试nonce，最终找到一个符合要求的nonce。在这种情况下，可以采用泊松分布进行近似，由此通过概率论可以推断出，系统出块时间服从指数分布。(需要注意的是，出块时间指的是整个系统出块时间，并非挖矿的个人)



系统平均出块时间为10min，该时间为系统本身设计，通过难度调整维护其平均出块时间。 指数分布本身也具有无记忆性。也就是说，对整个系统而言，已经过去10min，仍然没有人挖到区块，那么平均仍然还需要等10min（很不符合人的直觉）。也就是说，将来要挖多久和已经挖多久无关。

> 虽然这样看起来是一个冷酷的事情，过去的工作可能都会白做。但实际上这才是挖矿公平性的保障。对算力有优势的矿工来说，其之前所做大量工作仍有可能会白费。

比特币总量计算



也就是说，比特币系统中已经挖出和未挖出的比特币总数便是2100万个。 实际上，挖矿这一操作并非在解决数学难题，而是单纯的算力比拼。也就是说，挖矿这一过程并没有实际意义，但挖矿这一过程，却是对比特币系统的稳定起到重要维护作用。 所以，只要大多数算力掌握在好的节点手中，便能够保障比特币系统的稳定。

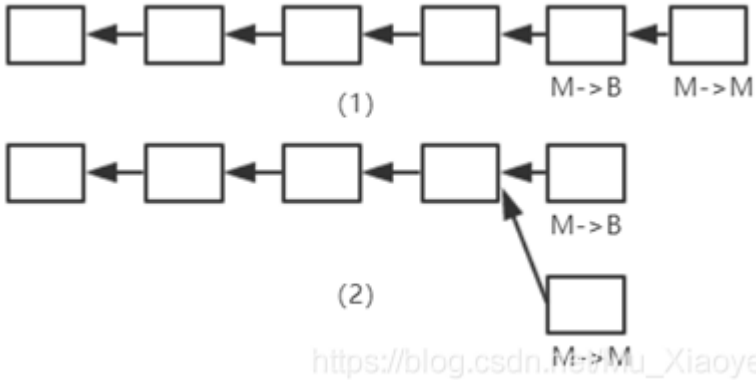
> 比特币越来越难被挖到，且出块奖励越来越少，是否说明其未来挖矿的动力将越来越低呢？ 实际上，恰恰相反。在早期比特币很容易挖到的时候，比特币并不被人们所看好，而后，比特币估值上涨，吸引其他人参与挖矿，又进一步促进了比特币价值上涨，进而又吸引更多人参与进来。 当出块奖励趋于0时，则整个系统将依赖于交易费运行，届时交易费将成为维护比特币系统运行的重要保障。

比特币系统安全性分析

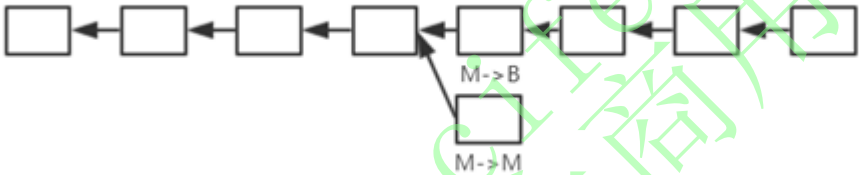
> 大多数算力掌握在好的用户手中，能否保障不良交易记录不会被写入区块链？ 需要注意的是，算力低的用户并非完全不能获得记账权，仅仅是概率上较低的问题。但实际上，即使拥有少量算力的恶意节点，也有一定概率获得某个区块的记账权。

- 1. 可否"偷币"？（恶意节点能不能将其他账户上比特币转给自己？） 答案：不能。因为转账交易需要签名，恶意节点无法伪造他人签名。加入其获得记账权并硬往区块中写入该交易，大多数用户会认为其是一个非法区块，大多数算力将不认可该区块，从而沿着其他路径挖矿，随着时间推移，拥有大多数算力的诚实的节点将会仍然沿着原来区块挖矿，从而形成一条"最长合法链"，该区块变成孤儿区块。对于攻击者来说，不仅不能偷到其他人的比特币，而且得不到出块奖励，还浪费了挖矿花费的电费等成本。
- 2. 可否将已经话过的币再花一遍？ 如下图1，若M已经将钱转给B，现在想再转给自己，假设其获得记账权，若按照图1方式，很明显为一个非法区块，不会被其他节点承认。 所以，M只能选择图2方式，将M转账给B的记录回滚掉。这样就有了两条等长合法链，取决于哪一个会胜出。（如果上面交易产生不可逆的外部效果，下面交易回滚便又拿回钱，从而不当获益）

> 需要注意的是，再挖矿之初便要选择上一个区块是谁。也就是说，并不是获得记账权之后才选择插入到哪一个区块之后。



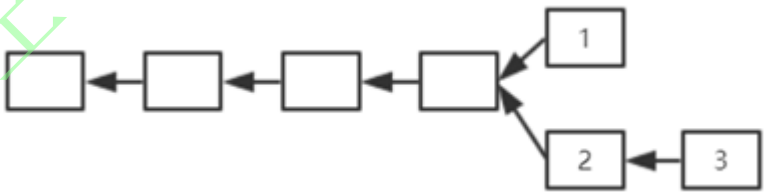
如何防范这种攻击？？？ 如果再M->B这个交易之后还延续有几个区块，如下图所示，则大多数诚实节点不会承认下面的链。所以，便变成了恶意节点挖下面的链，其他节点挖上面的链的算力比拼。由于区块链中大多数节点为善意节点，则最终上面链会胜出，而恶意节点的链会不被认可，从而导致投入成本白费。



所以，一种简单防范防范便是多等几个确认区块。比特币协议中，缺省需要等6个确认区块，此时才认为该记录是不可篡改的。平均出块时间10min，六个确认区块便需要1小时，可见等待时间还是相对较长的。

- 3. 可否故意不包含合法交易？ 可以，但是可以等待后续区块包含，所以问题不大。实际运行中，可能由于某段时间实际交易数太多，而一个区块包含交易数存在最大值，导致某些合法交易并未被写入区块链（等待后续区块写入）。
- 4. selfish mining 提前挖到但不发布，继续挖下去，等到想要攻击的交易等了6次确认认为安全之后将整条链发布出去，试图回滚原来记录。这种情况，需要恶意节点掌握系统中半数以上算力才行，否则无法成为最长合法链。

> selfish mining有好处吗？ 如图所示，假使挖到2号时候先不发布，则其他人仍然需要挖1号区块，若其算力足够强，能保证别人挖出1之后可以挖出3.可以此时将2和3一起发布，从而将1区块所在链最长合法链挤掉（减少了别人和自己竞争挖3号区块）。 但这样存在风险，如果别人已经挖出1，自己还没挖出3，则需要尽快发布2和别人竞争最长合法链地位。



需要注意的是，比特币系统中，假如发生以下情况，各个节点以自己先收到的区块所在链为主链，对后收到的合法区块会不予认可（但会先保存起来）。此时便变成了两批算力分布挖1和2，具体哪一个成为主链，取决于哪一条链先挖到下一个区块，使得两个等长合法链出现长短不一致，最终胜者成为最长合法链。