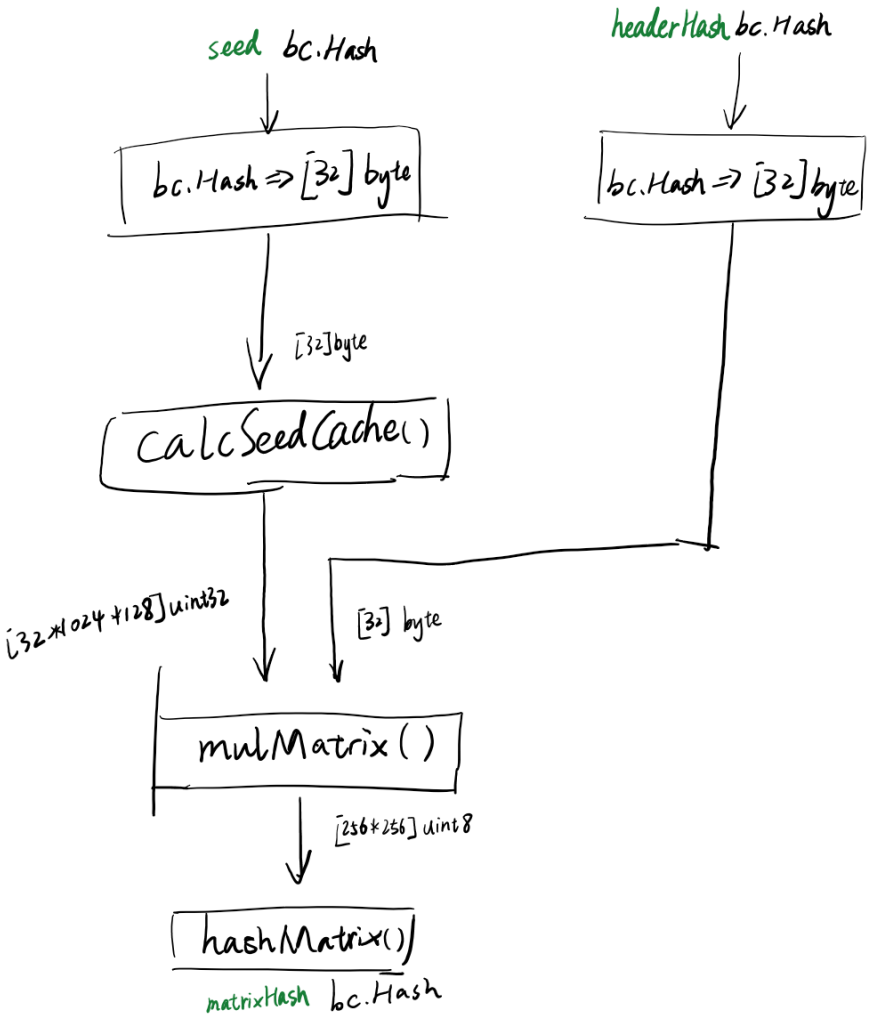
```
func Hash(hash, seed *bc.Hash) *bc.Hash
```

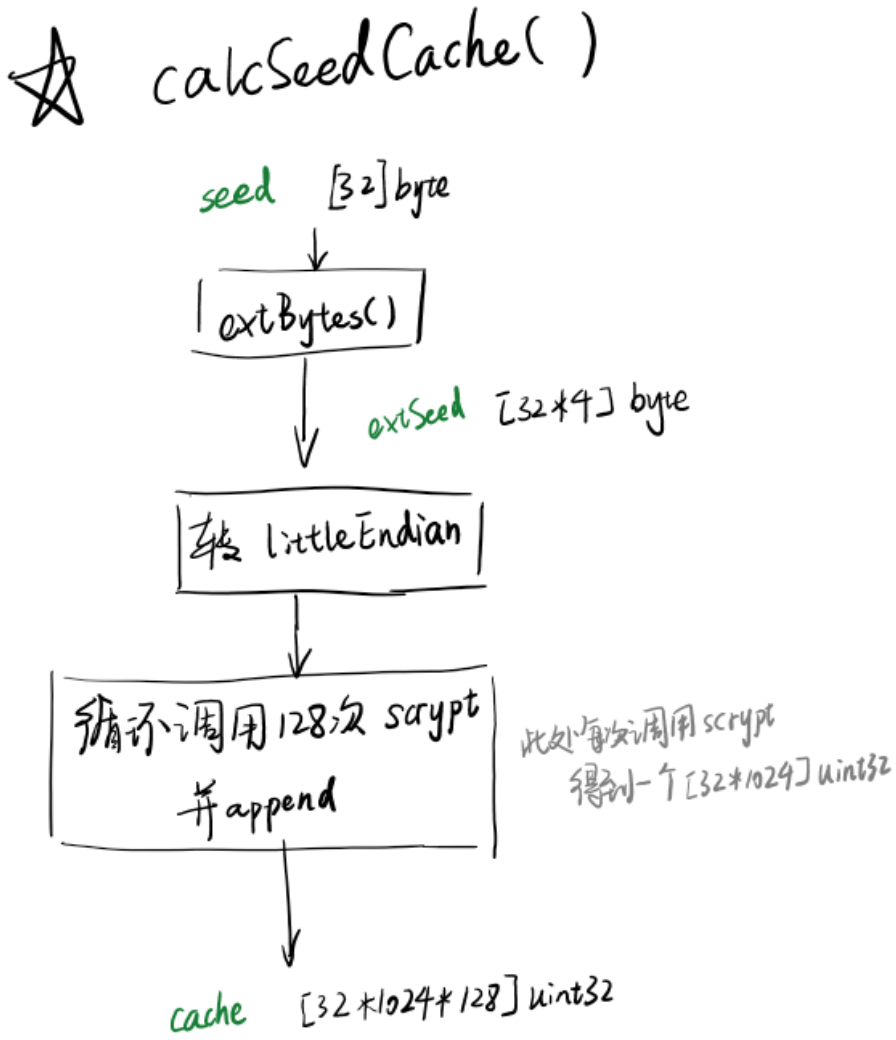
☆ Hash( )

☆ bc.Hash 是个结构体，含4个 fixed64，即 4 * 64 bit

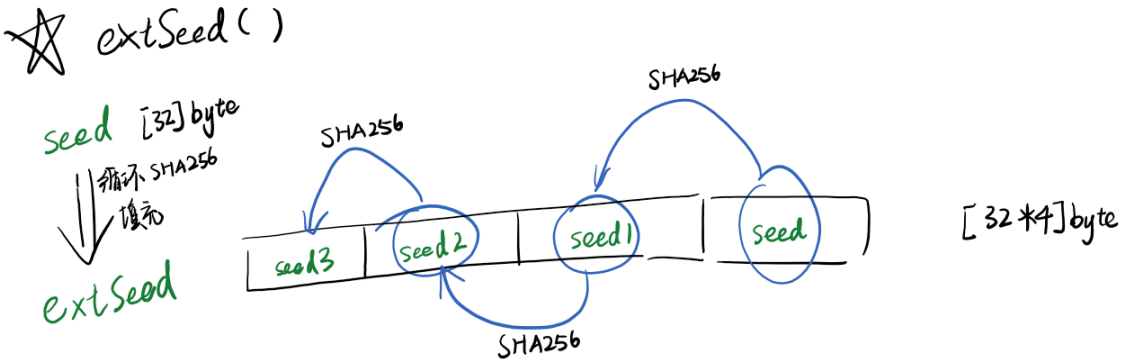
☆ headerHash 及 seed 由外部传入，其中，headerHash 充当被 hash 压的 nounce，seed 则为 ???



```go
func calcSeedCache(seed []byte) (cache []uint32)
```




```go
func extendBytes(seed []byte, round int) []byte
```

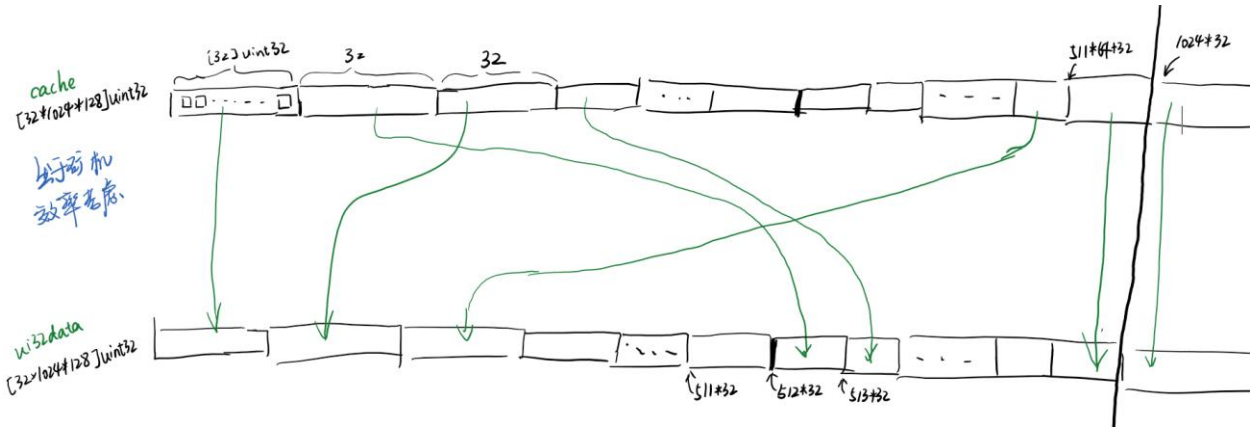


```
func mulMatrix(headerhash []byte, cache []uint32) []uint8
```

IN: headerhash [32] byte

cache [32*1024*128] uint32

OUT: result [256*256] uint8



i8data [256*256*256] int8

f64data [256*256*256] float64

dataIdentity

[256*256]float64

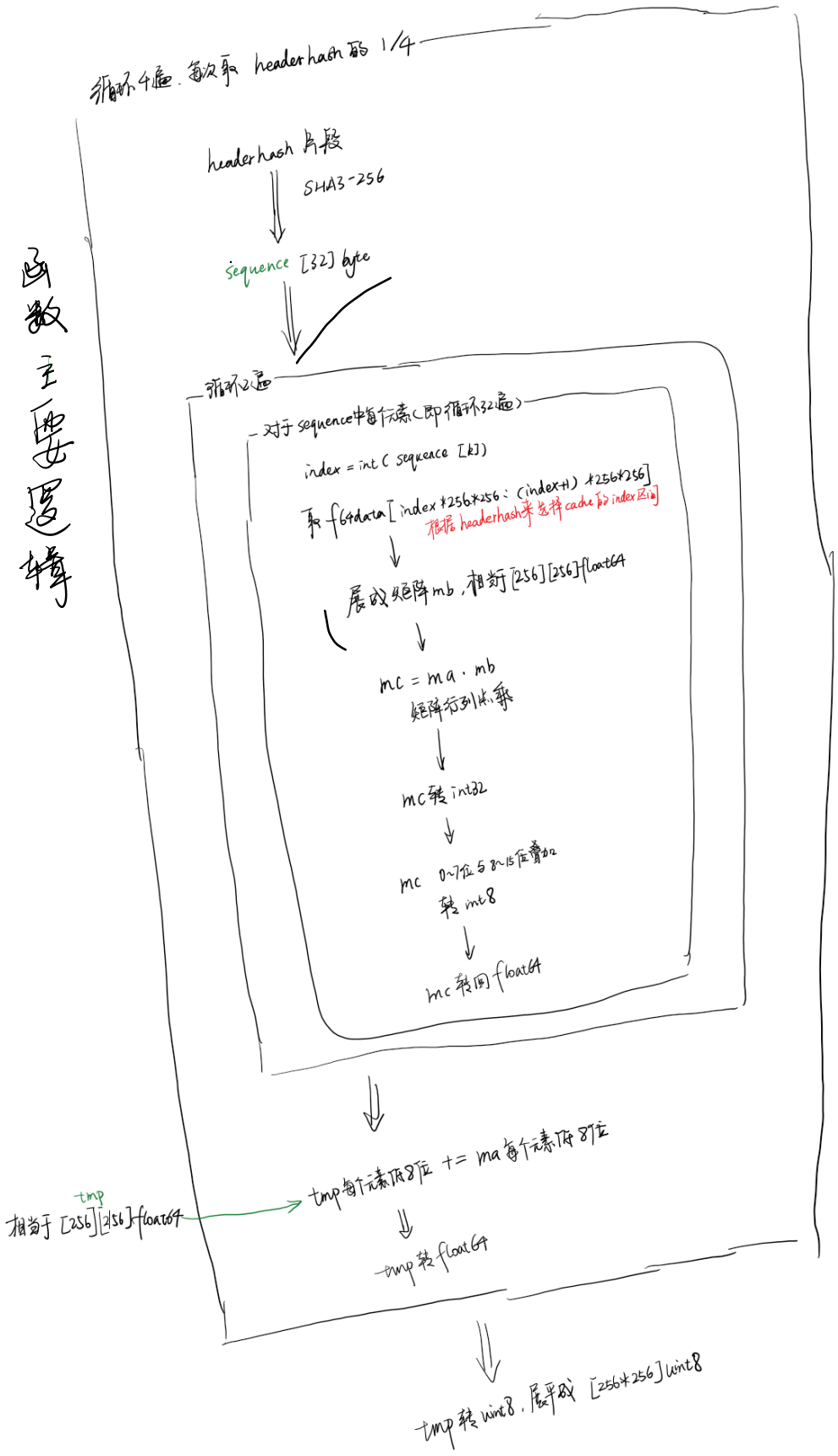
从0开始，每 i*257 置 1 (即每 256 个，offset 自增，((i*256)+offset)处置1)

0 ~ 255*257 = (256-1)(256+1) = $256^2 - 1$

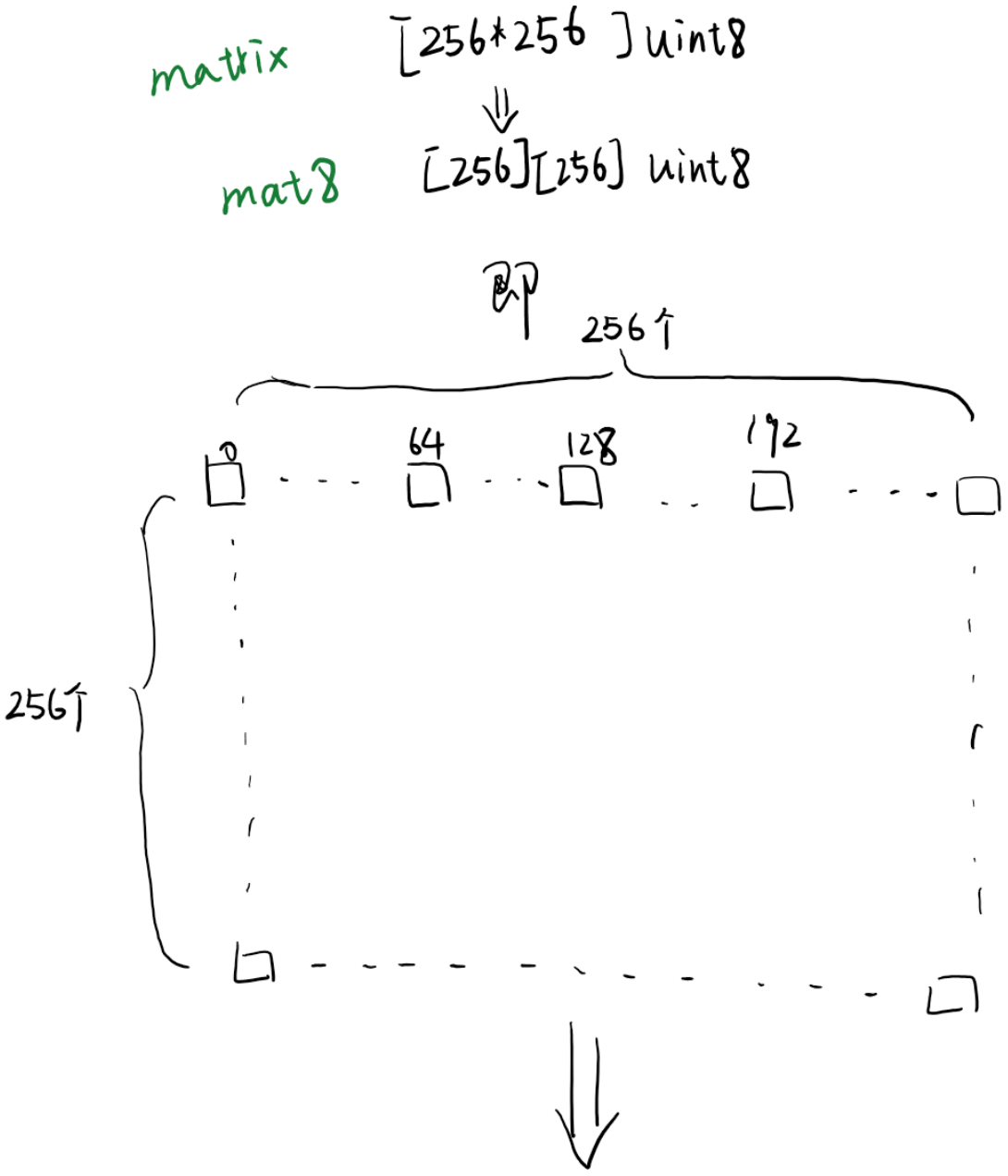
⇓

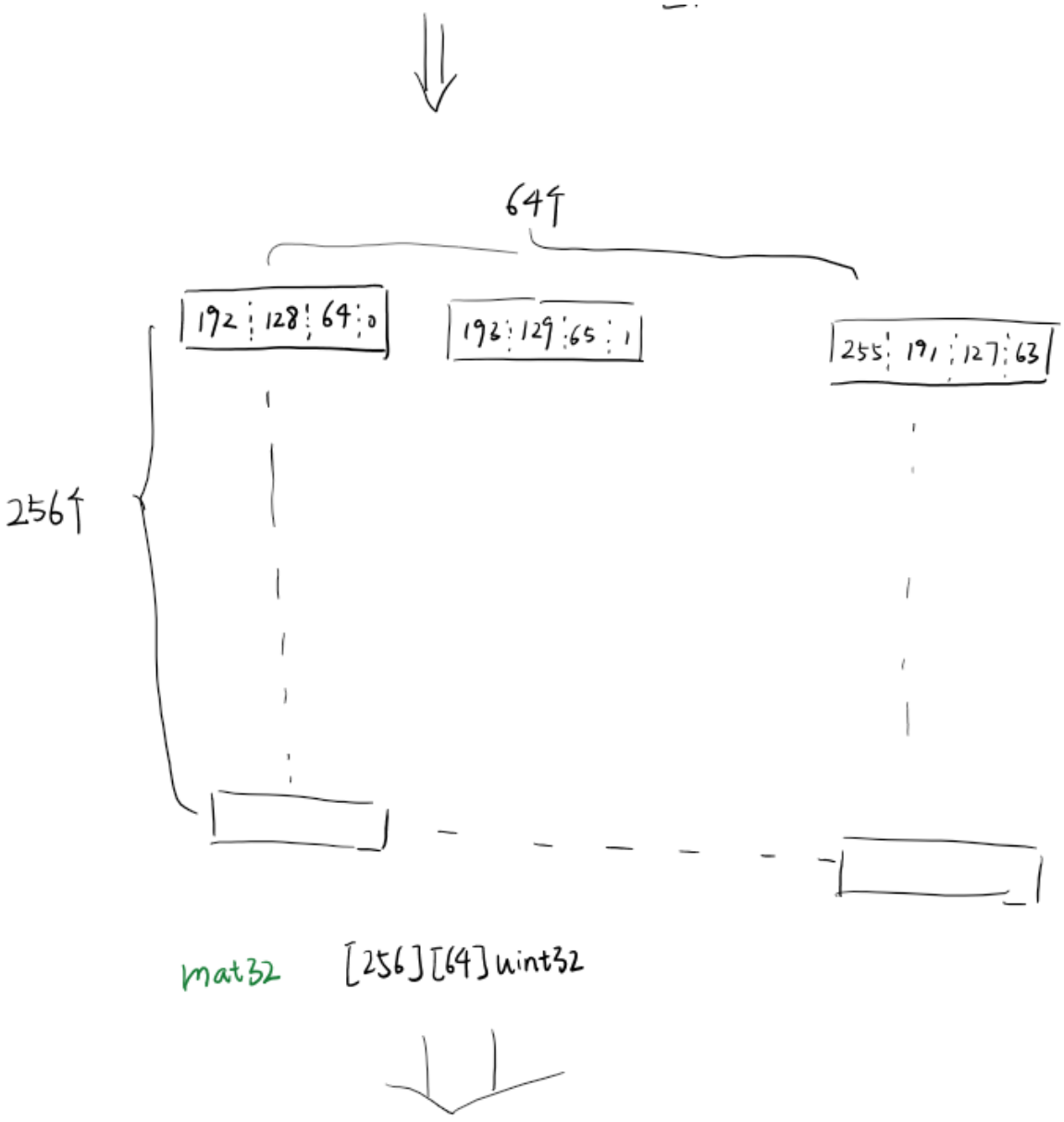
相当于 [256][256]float64

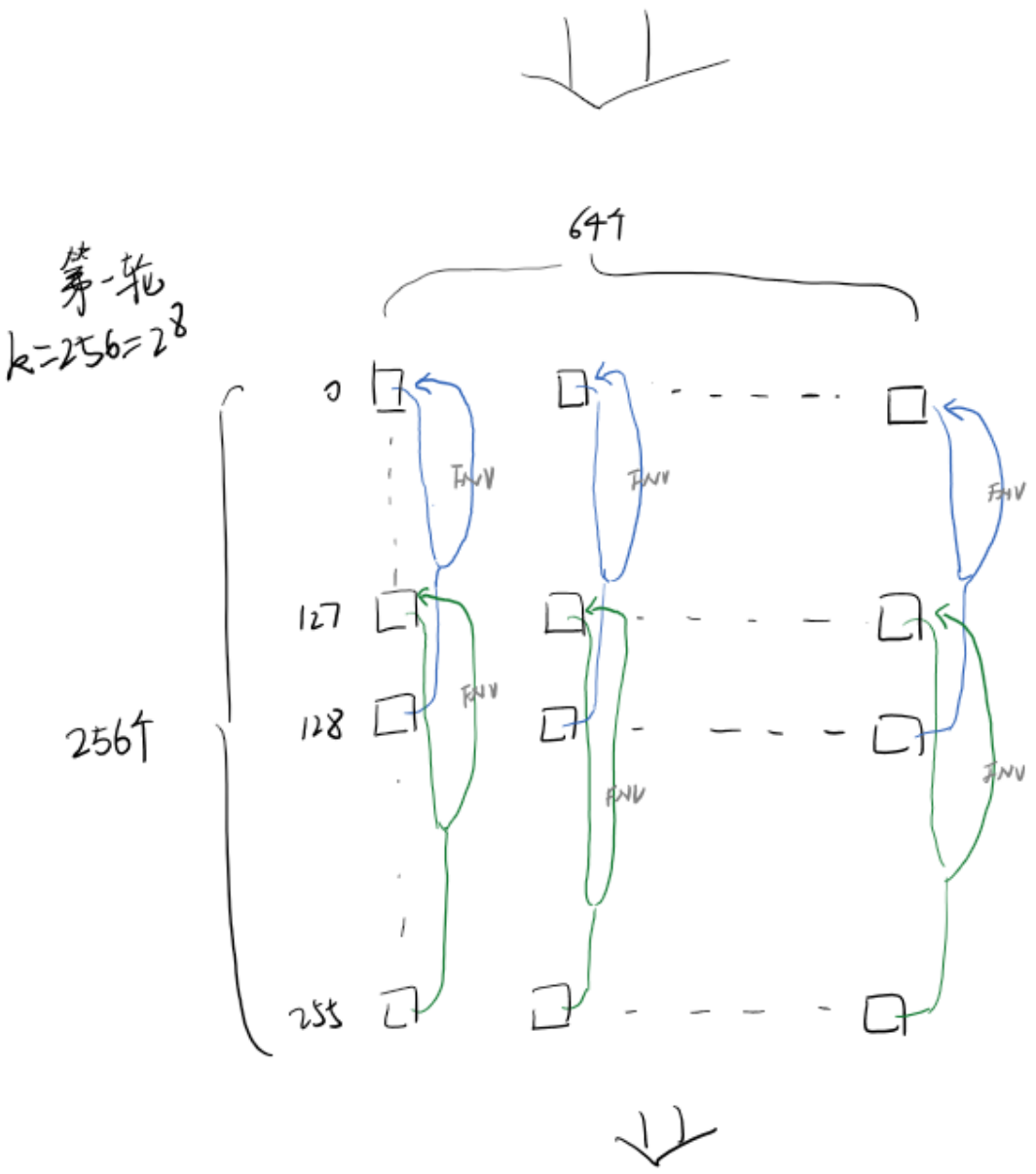
ma

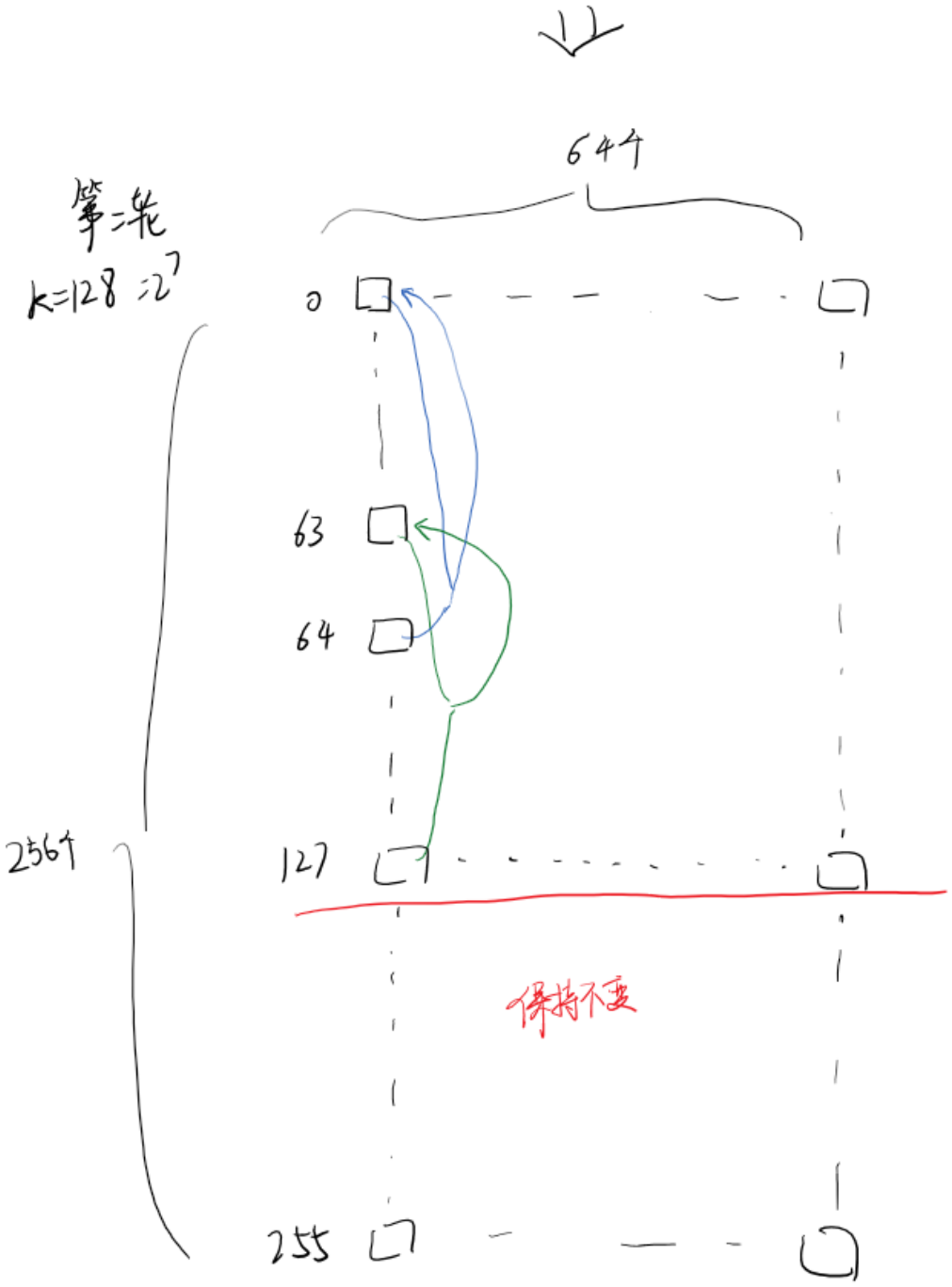

函数主要逻辑
循环4遍，每次取 header hash 的 1/4
header hash 为段
SHA3-256
sequence [32] byte
循环2遍
对于 sequence中每个元素(即循环32遍)
index = int(sequence [k])
取 f64data[index*256*256 : (index+1) *256*256]
根据 header hash来选择 cache的 index区间
展成矩阵 mb，相当于[256][256]float64
mc = ma · mb
矩阵行列点乘
mc转 int32
mc 0~7位与 8~15位异或
转 int8
mc转回 float64
tmp每个元素加8位 += ma每个元素加8位
tmp转 float64
相当于 [256][256]float64
tmp
tmp转 uint8，展平成 [256*256] uint8

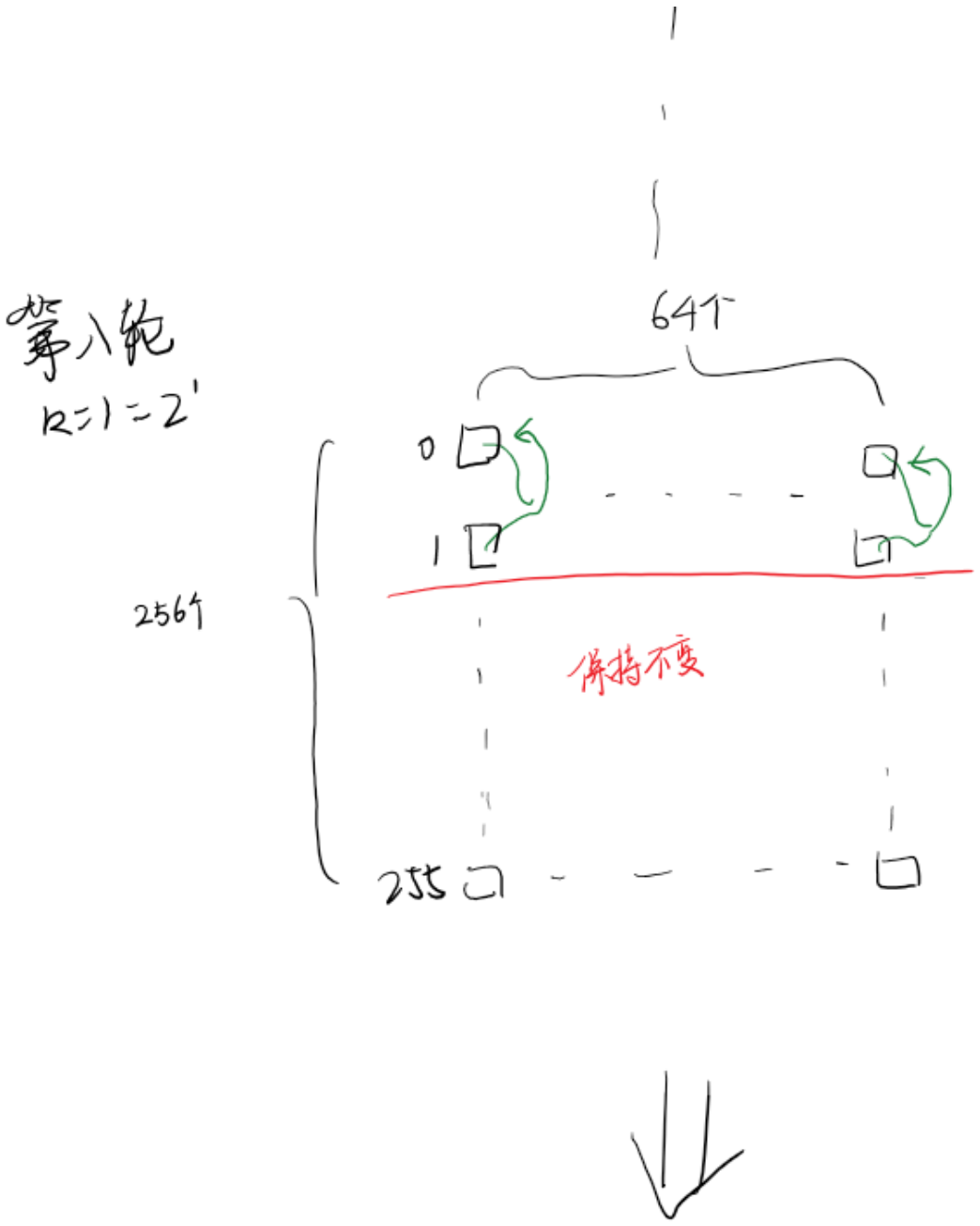
```
func hashMatrix(result []uint8) *bc.Hash
```




64↑
192 128 64 0
193 129 65 1
255 191 127 63
256↑
mat32 [256][64]uint32


第一轮
k=256=2^8
64个
256个
0
127
128
255
FNV
FNV
FNV
FNV
FNV


64个
第二张
k=128 :2^7
0
63
64
127
255
256个
保持不变


等八轮
K=1=2'
64个
0
1
256个
保持不变
255

ui32data [64]uint32

(取矩阵第一行)

dataBytes [256] byte

SHA3-256

[32] Byte

bc.NewHash()

bc.Hash (4*fixed64)