Merkle trees

Flag Bits

Use of Merkle trees

- Create and verify an inclusion proof
- I.e. I want a compact proof that one or more data belongs to a Merkle tree with a specific Merkle root

- Important: one can get proofs for multiple data points

- Recall: everyting in a Merkle tree is a hash (modulo leaves; sort of)

What does a proof look like?

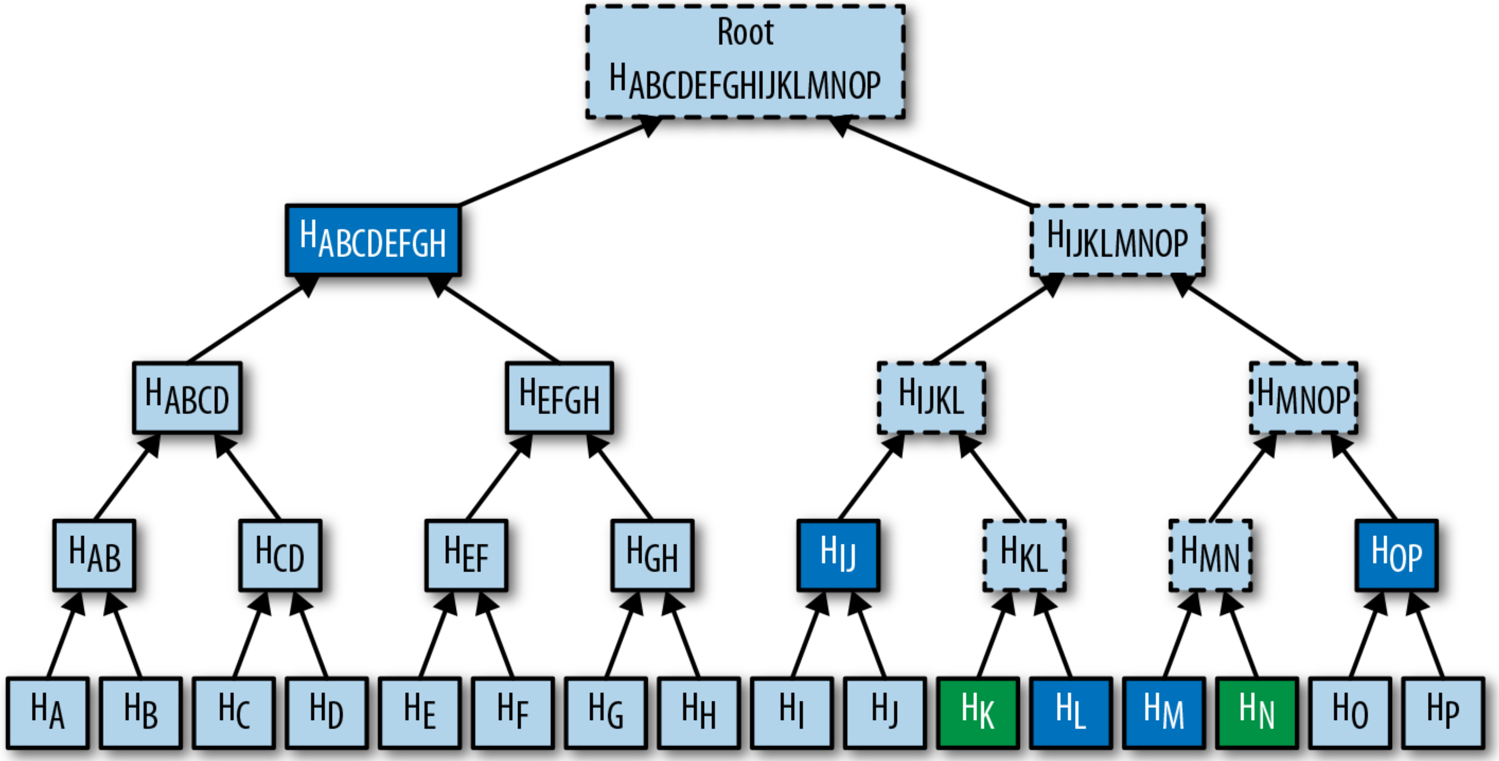


Image from Jimmy Song: https://github.com/jimmysong/programmingbitcoin/blob/master/ch11.asciidoc

What does a proof look like?

I want a Proof that:
- H(K) and H(N) belong
- To the tree with root H(A-P)

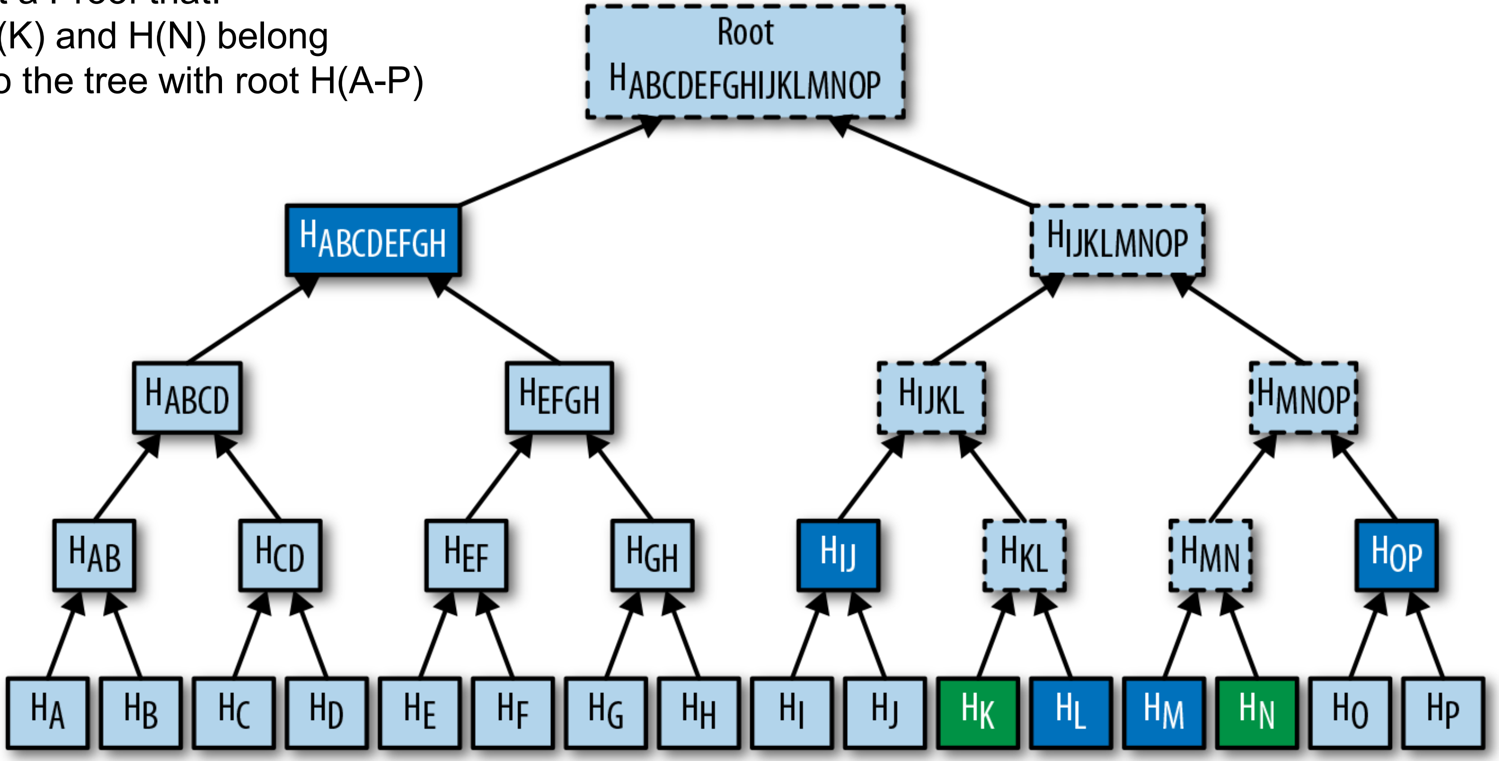
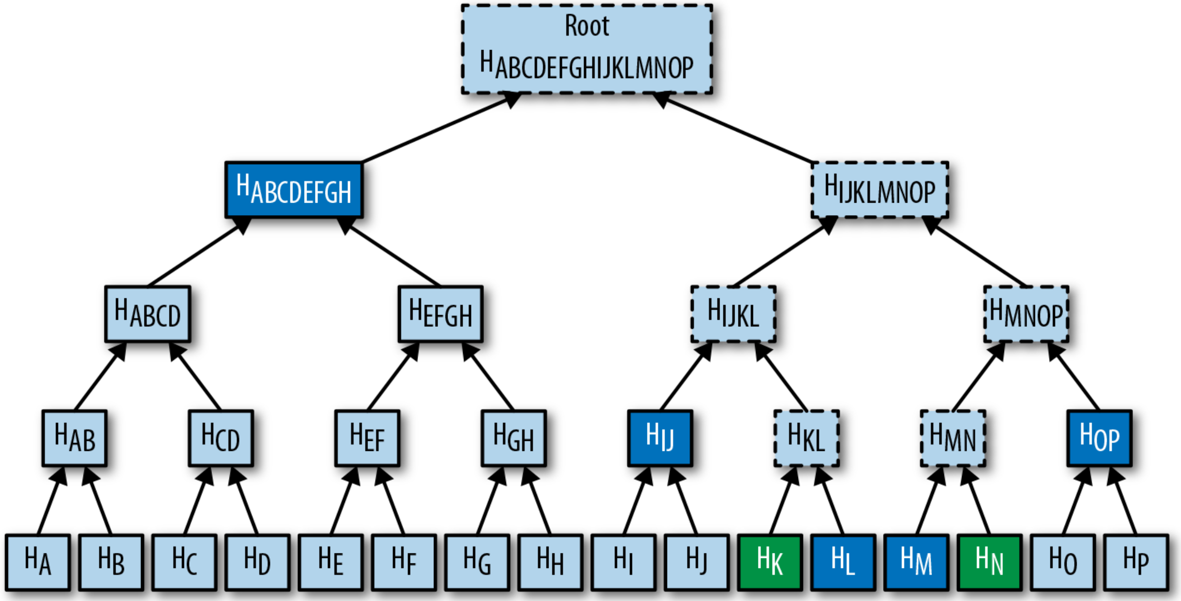


Image from Jimmy Song: https://github.com/jimmysong/programmingbitcoin/blob/master/ch11.asciidoc

What does a proof look like?

I want a Proof that:
- H(K) and H(N) belong
- To the tree with root H(A-P)

I need the blue hashes:
- H(L) to compute H(KL)
- H(M) for H(MN)
- H(OP) for H(MNOP)
- H(IJ) for H(IJKL)
- H(A-H) for the root

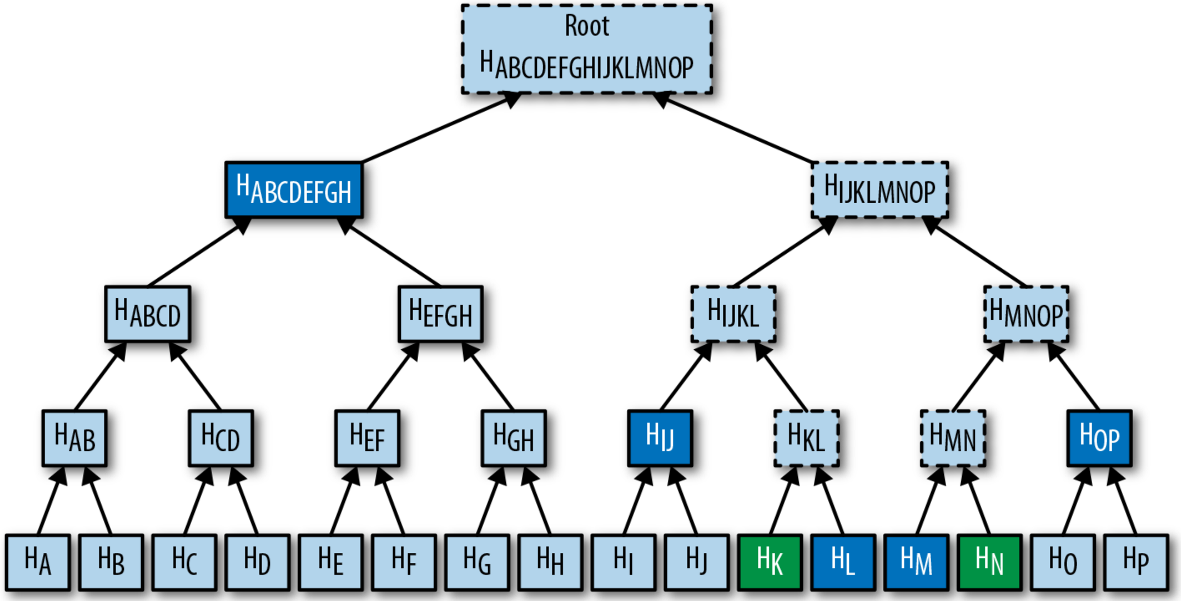


What does a proof look like?

I want a Proof that:
- H(K) and H(N) belong
- To the tree with root H(A-P)

I need the blue hashes:
- H(L) to compute H(KL)
- H(M) for H(MN)
- H(OP) for H(MNOP)
- H(IJ) for H(IJKL)
- H(A-H) for the root

Is this enough?

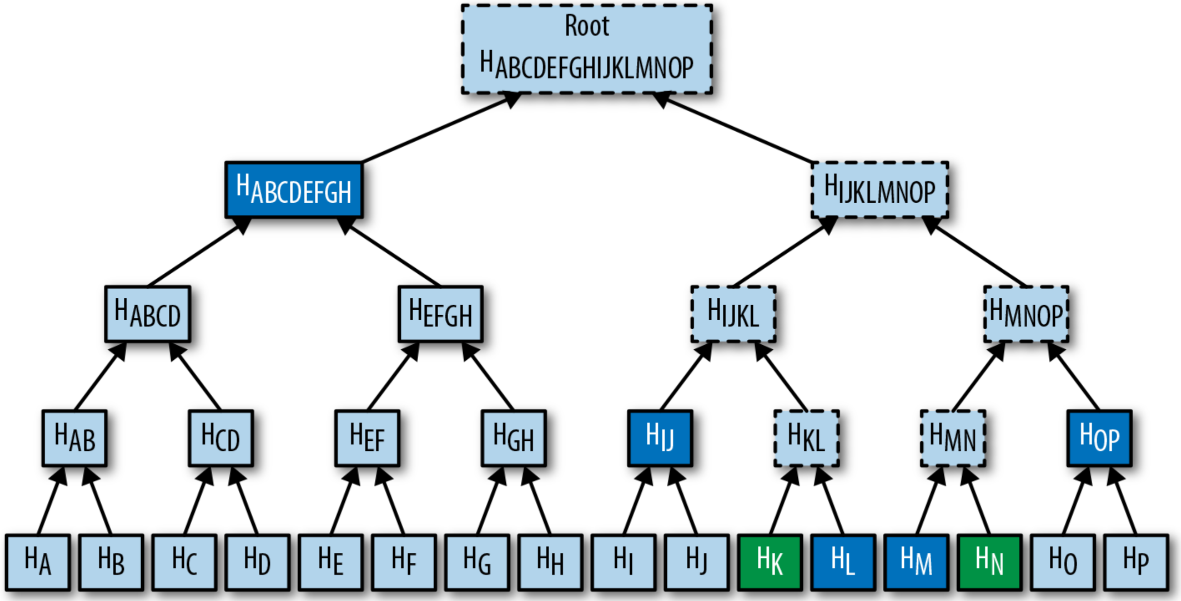


What does a proof look like?

I want a Proof that:
- H(K) and H(N) belong
- To the tree with root H(A-P)

I need the blue hashes:
- H(L) to compute H(KL)
- H(M) for H(MN)
- H(OP) for H(MNOP)
- H(IJ) for H(IJKL)
- H(A-H) for the root

Is this enough?



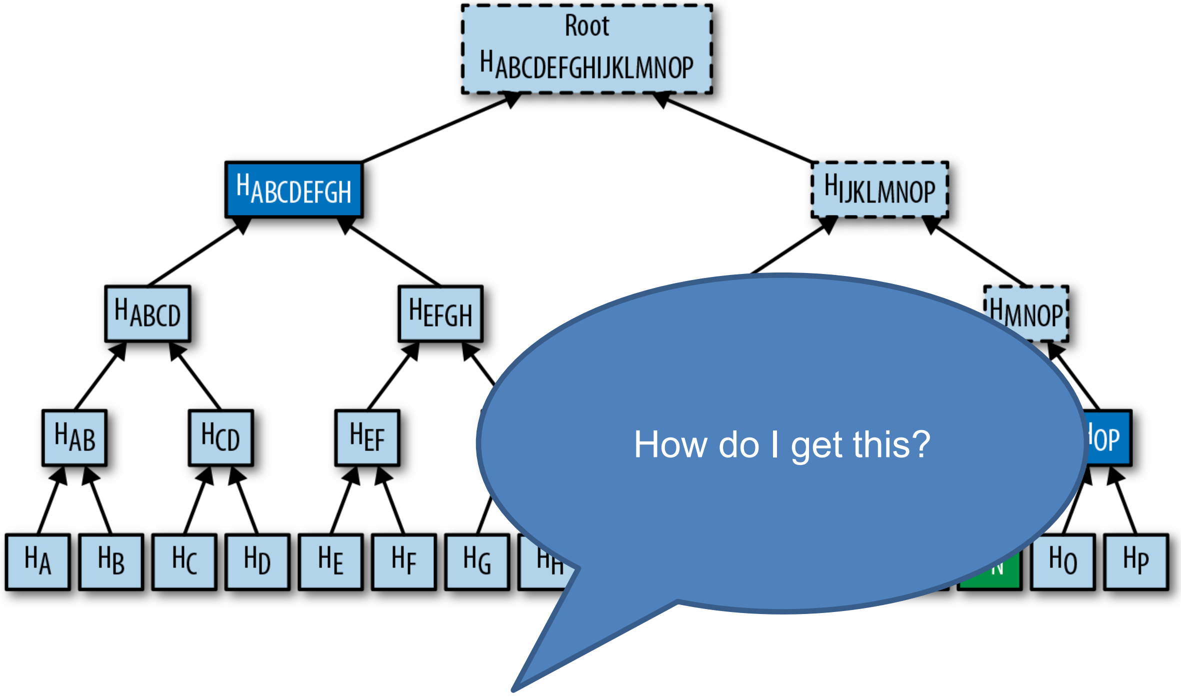
I also need: the position of all blue/green hashes in the tree!!!

What does a proof look like?

I want a Proof that:
- H(K) and H(N) belong
- To the tree with root H(A-P)

I need the blue hashes:
- H(L) to compute H(KL)
- H(M) for H(MN)
- H(OP) for H(MNOP)
- H(IJ) for H(IJKL)
- H(A-H) for the root

Is this enough?



I also need: the position of all blue/green hashes in the tree!!!

Tree traversal algorithms

BFS

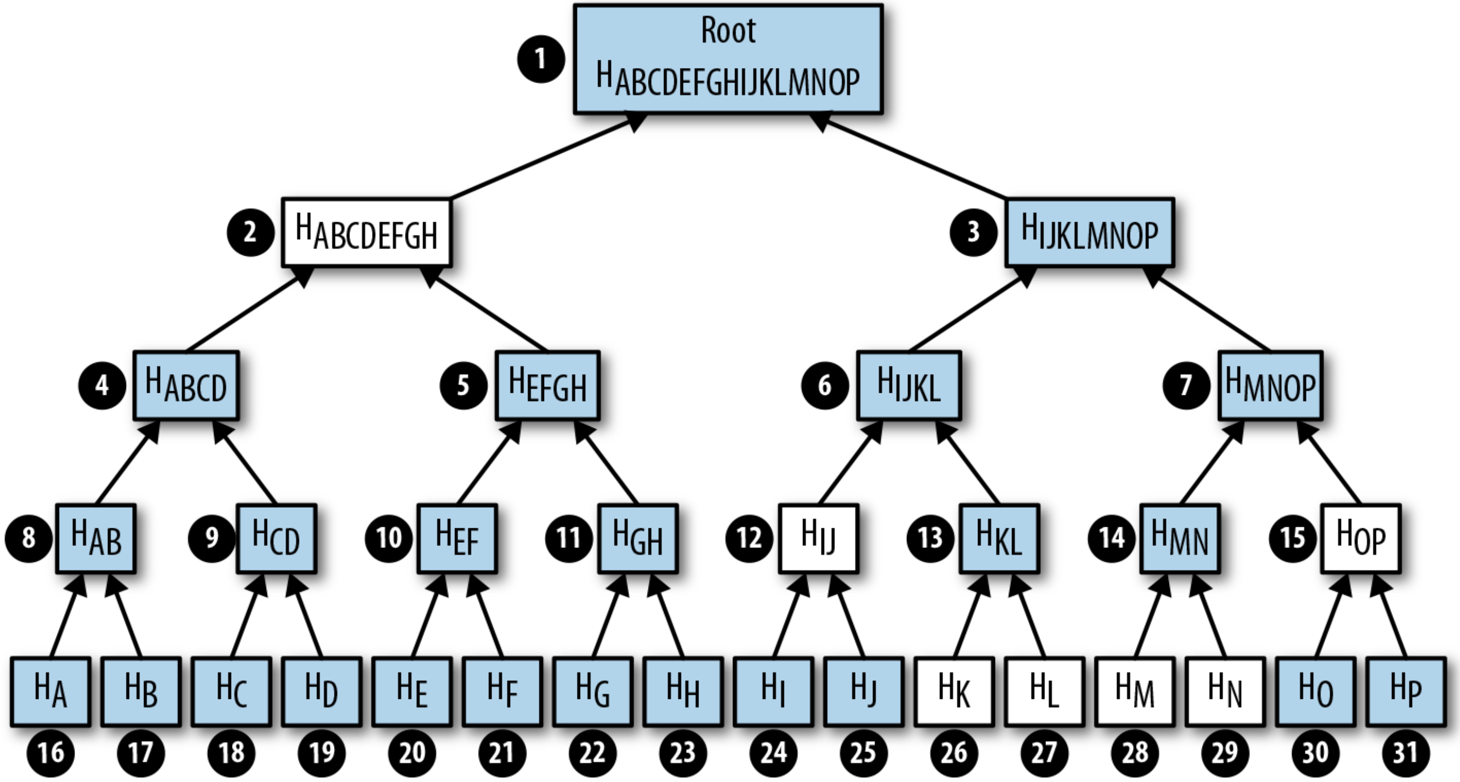


Image from Jimmy Song: https://github.com/jimmysong/programmingbitcoin/blob/master/ch11.asciidoc

Tree traversal algorithms

DFS

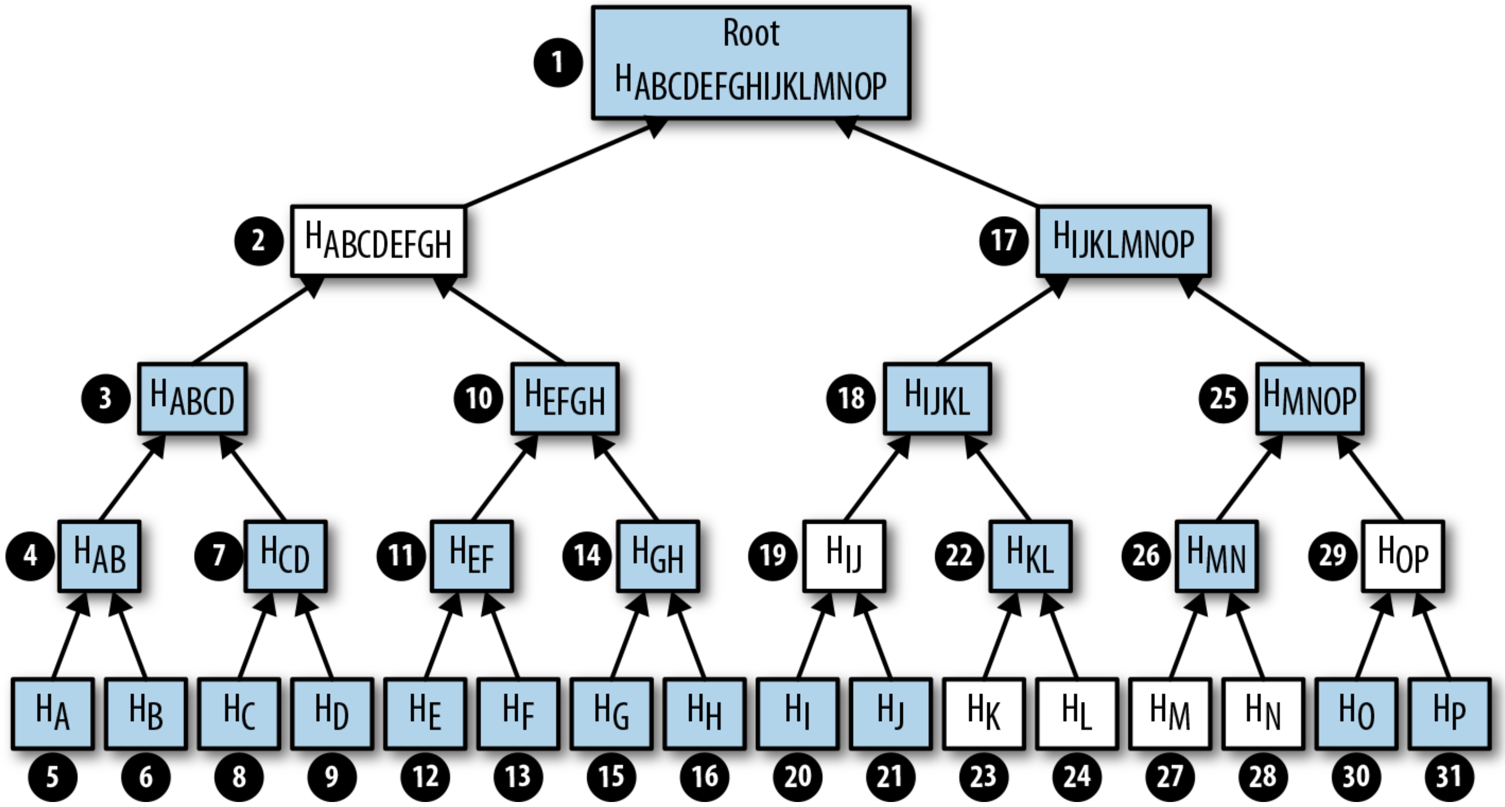


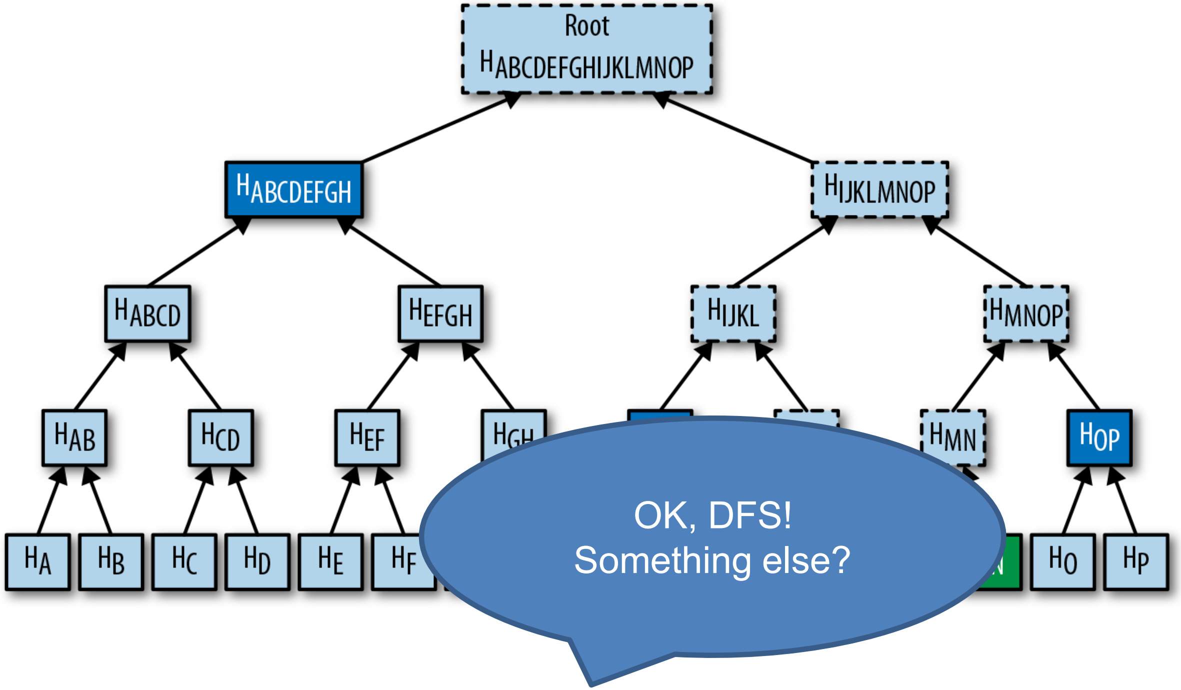
Image from Jimmy Song: https://github.com/jimmysong/programmingbitcoin/blob/master/ch11.asciidoc

What does a proof look like?

I want a Proof that:
- H(K) and H(N) belong
- To the tree with root H(A-P)

I need the blue hashes:
- H(L) to compute H(KL)
- H(M) for H(MN)
- H(OP) for H(MNOP)
- H(IJ) for H(IJKL)
- H(A-H) for the root

Is this enough?



I also need: the position of all blue/green hashes in the tree!!!

One last thing: the number of leaves!

What is done in Bitcoin?

- Proof that hashesOfInterest belong to Merkle tree with the root *Mroot:*

1) Numer of leaves in the tree with root Mroot
2) Hashes needed to reconstruct the root
3) Position of these hashes in the tree (using DFS)

What is done in Bitcoin?

- Proof that hashesOfInterest belong to Merkle tree with the root *Mroot:*

1) Numer of leaves in the tree with root Mroot
2) Hashes needed to reconstruct the root
3) Position of these hashes in the tree (using DFS)

    - How to represent this information?
    - With a list of bits!!!
    - **Flag Bits**

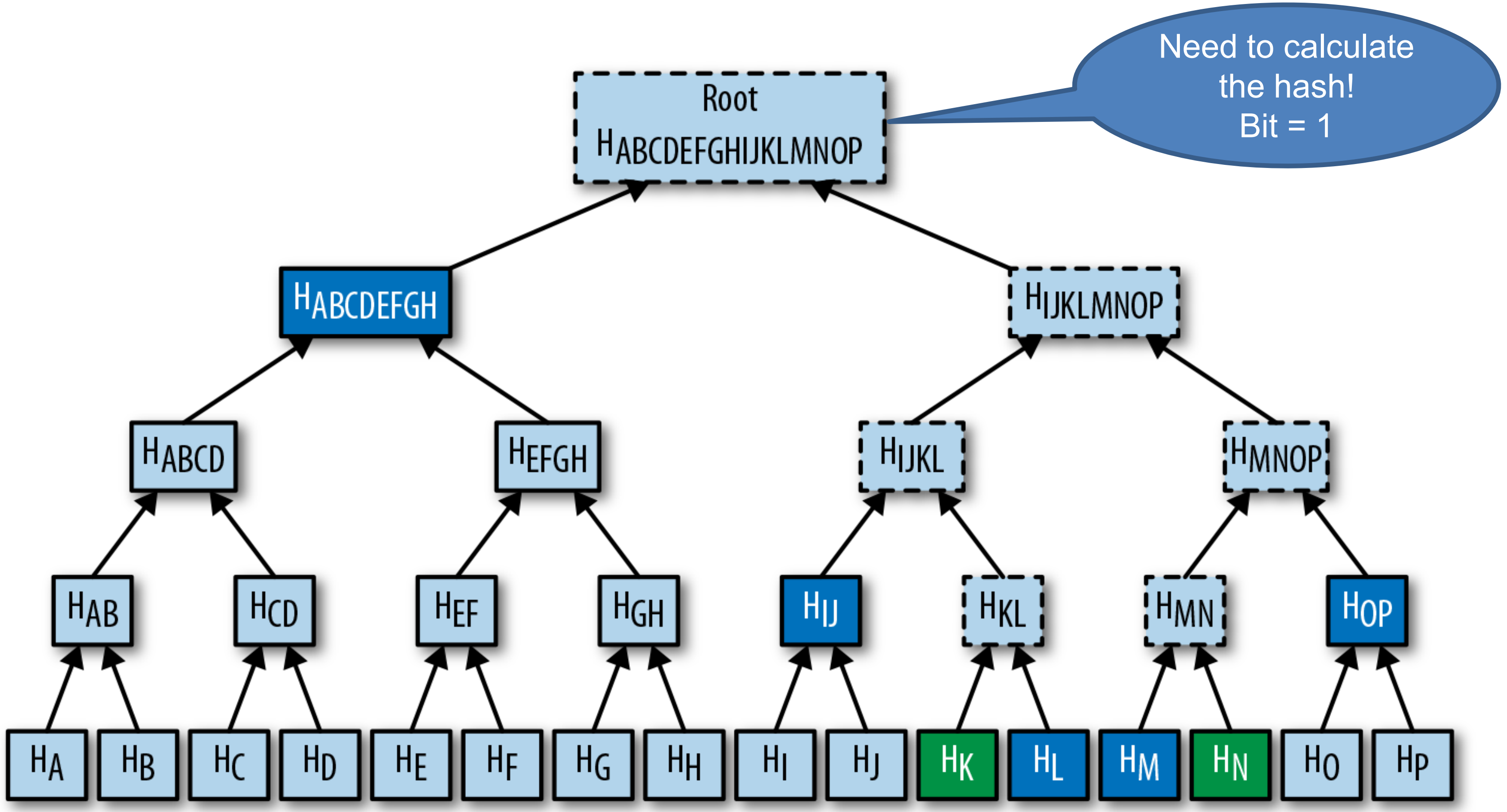
Flag Bits

Proof = (nrLeaves, listOfHashes, flagBits) for hashesOfInterest

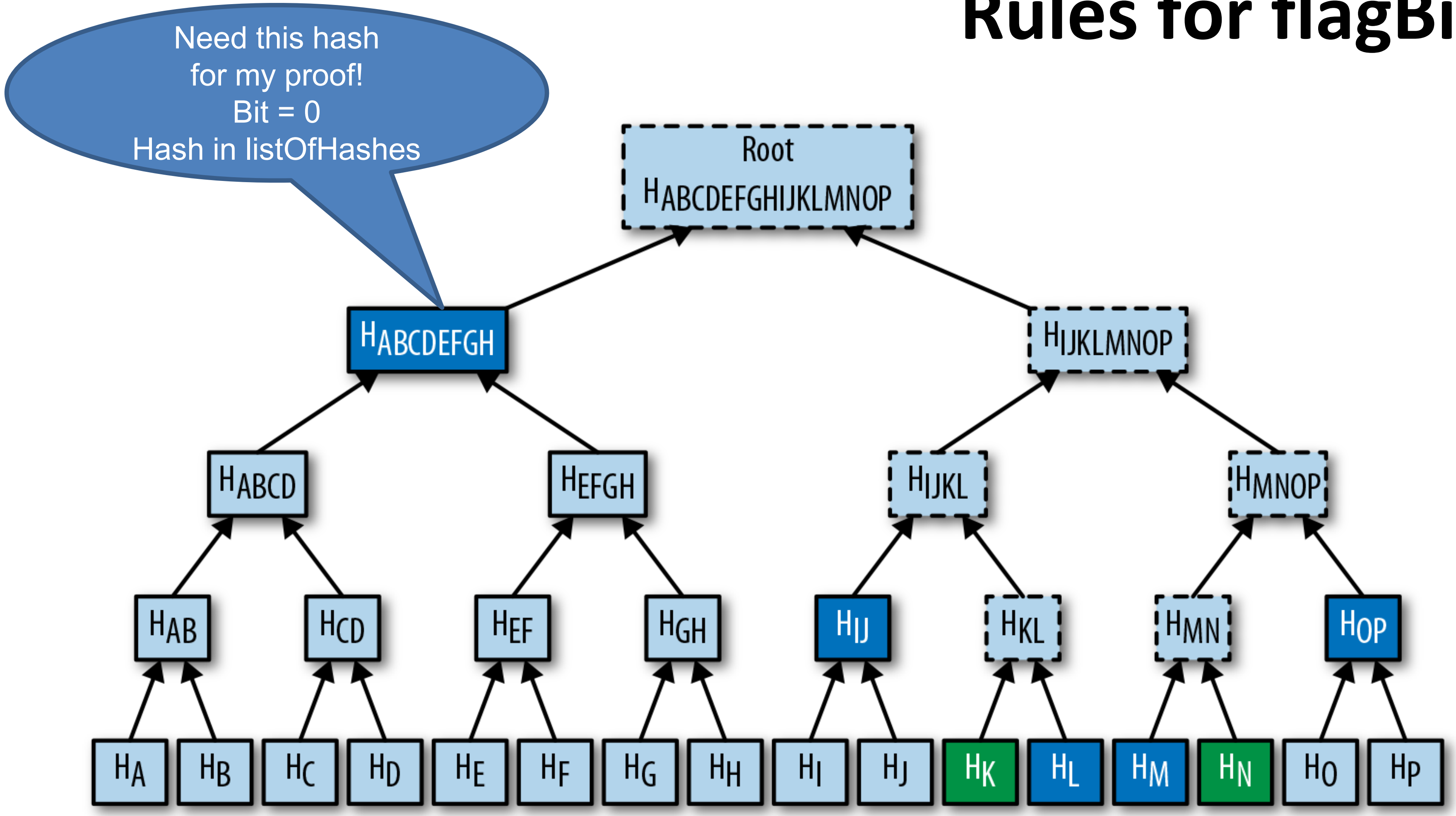
- Rules for flagBits:

1. If the hash of the node we are processing appears in listOfHashes, the bit is equal to 0
2. If the node is an internal node (not a leaf), and we need to compute its value, the bit equals 1
3. If the node is a leaf that appears in hashesOfInterest, then the bit equals to 1, and the hash appears in the listOfHashes

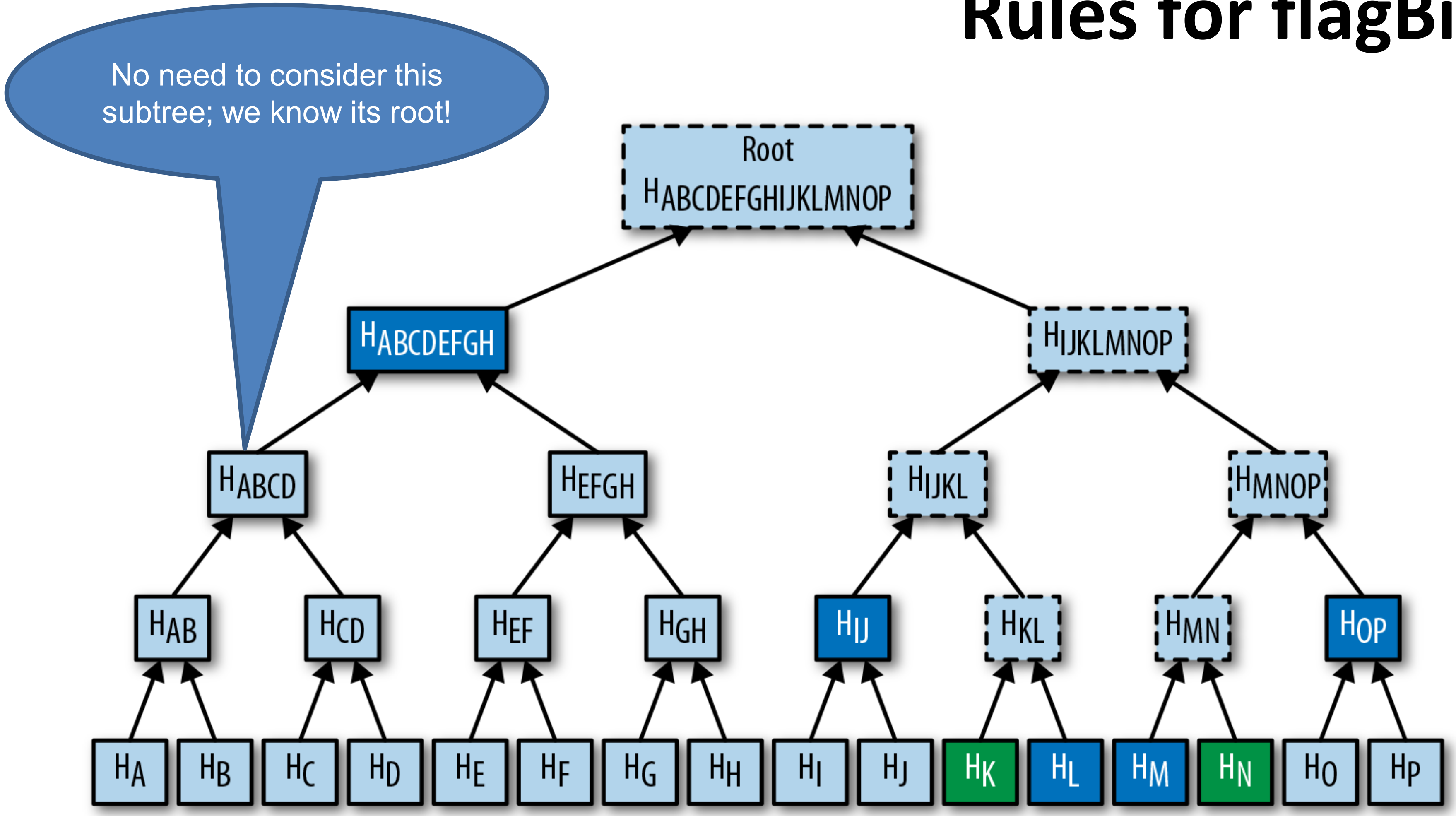
Rules for flagBits



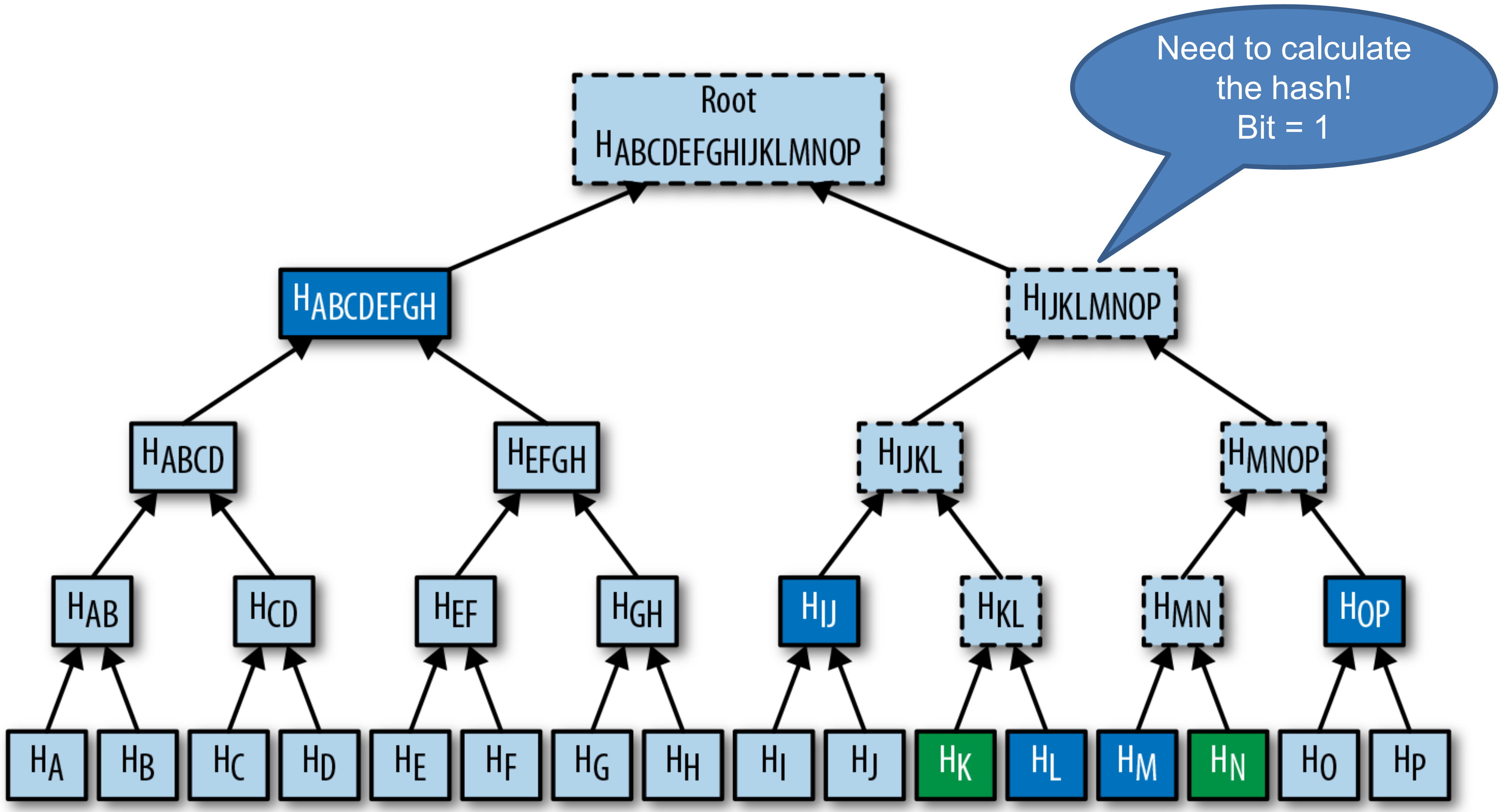
Rules for flagBits



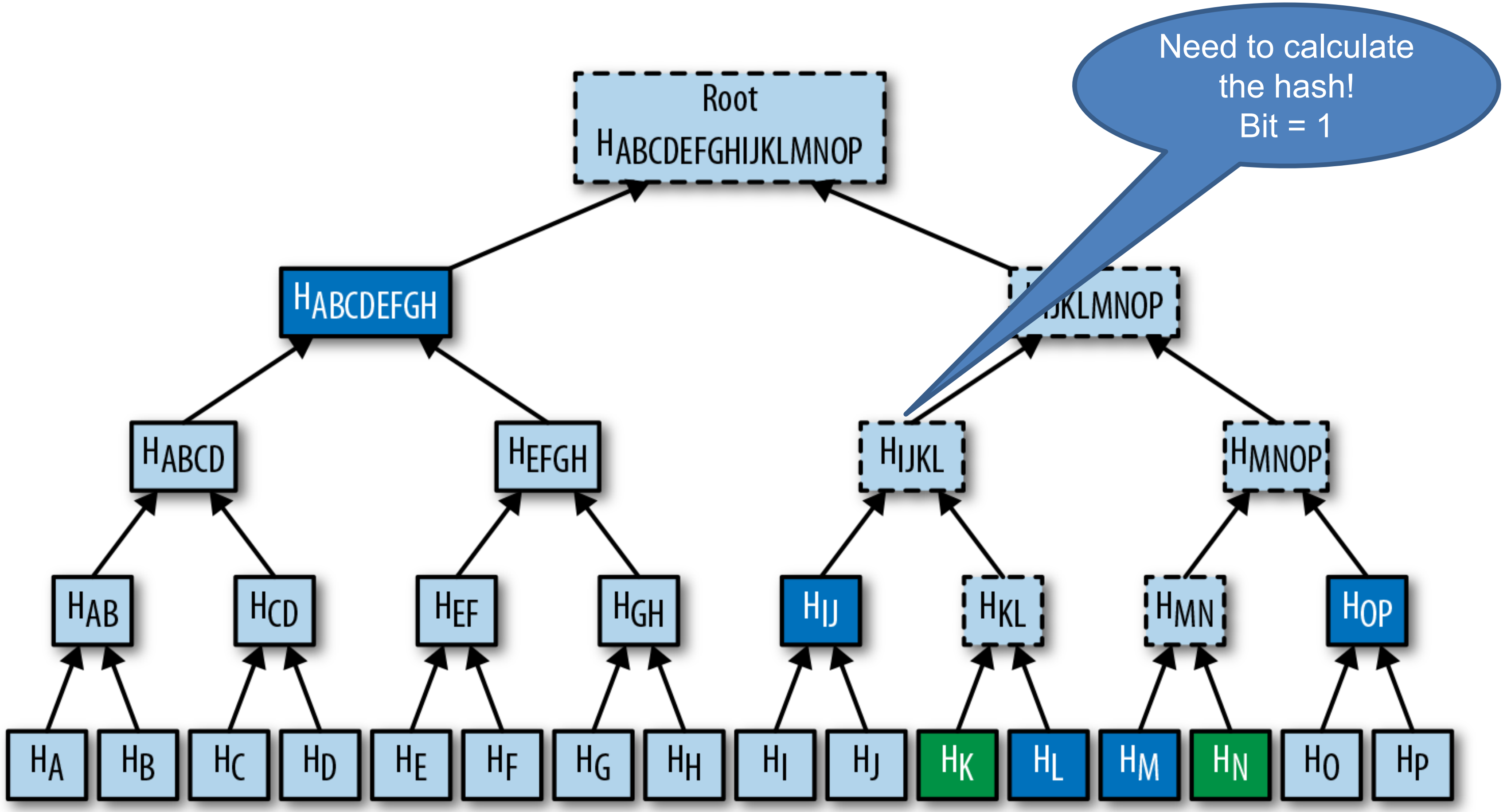
Rules for flagBits



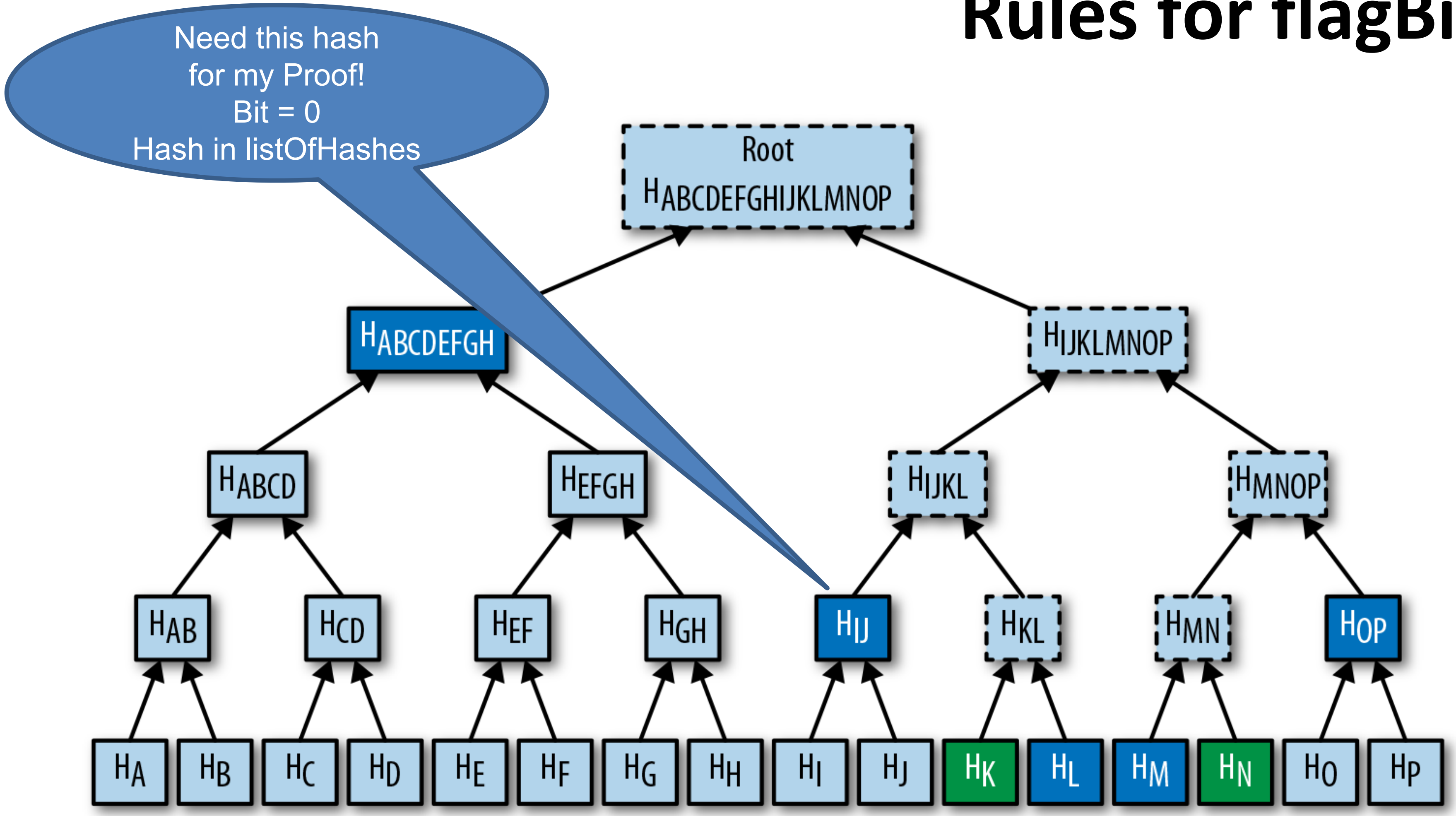
Rules for flagBits



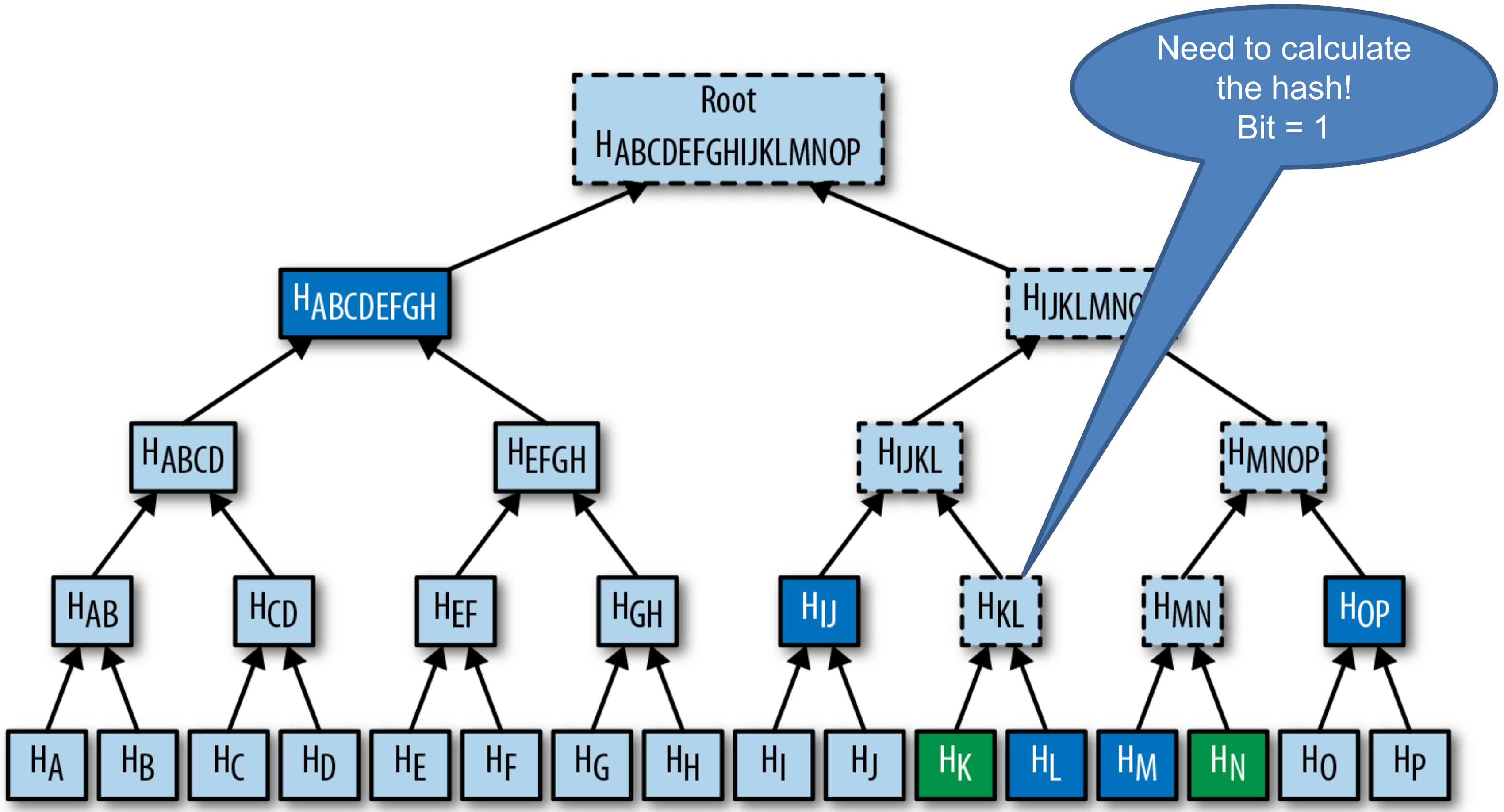
Rules for flagBits



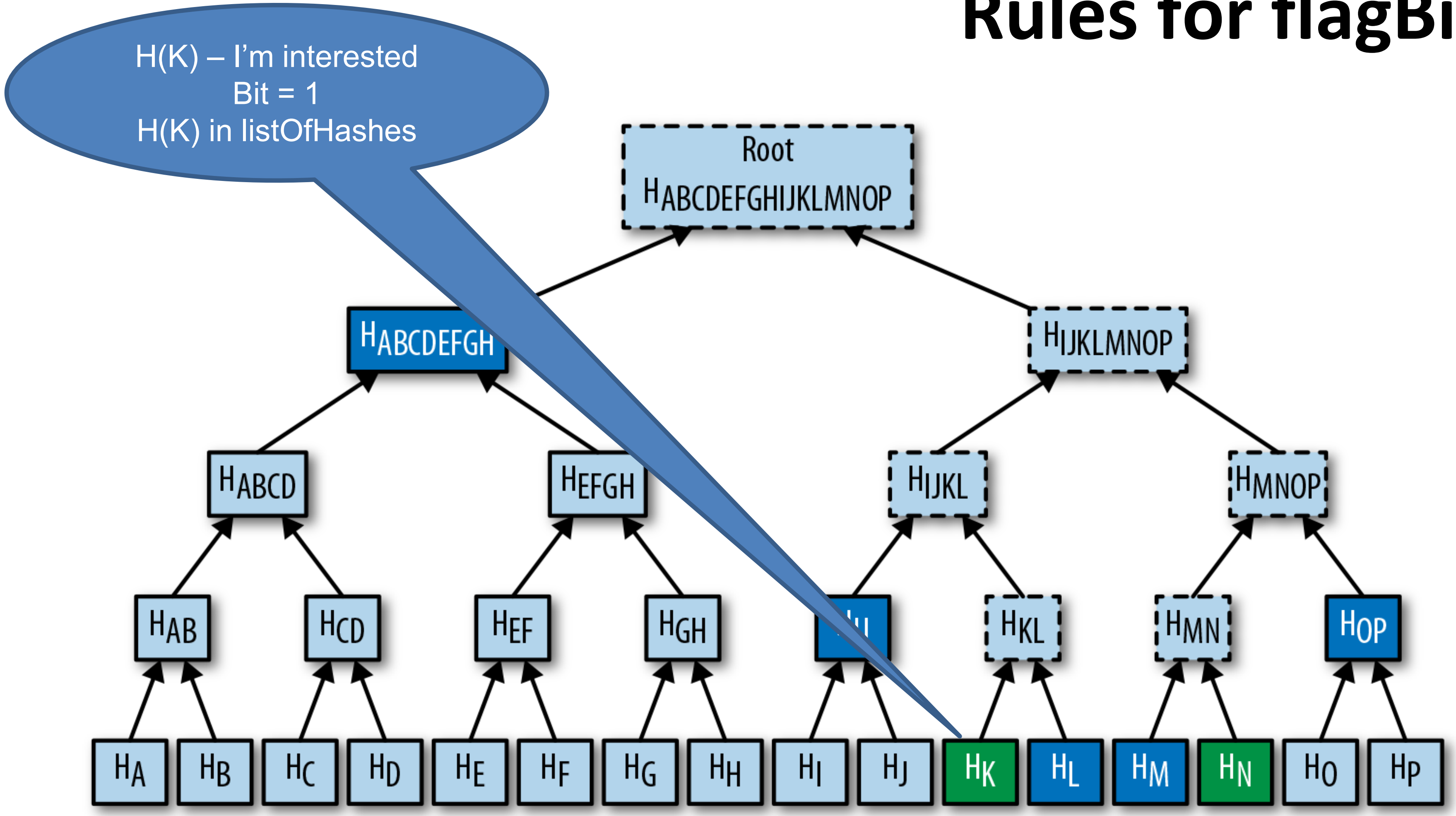
Rules for flagBits



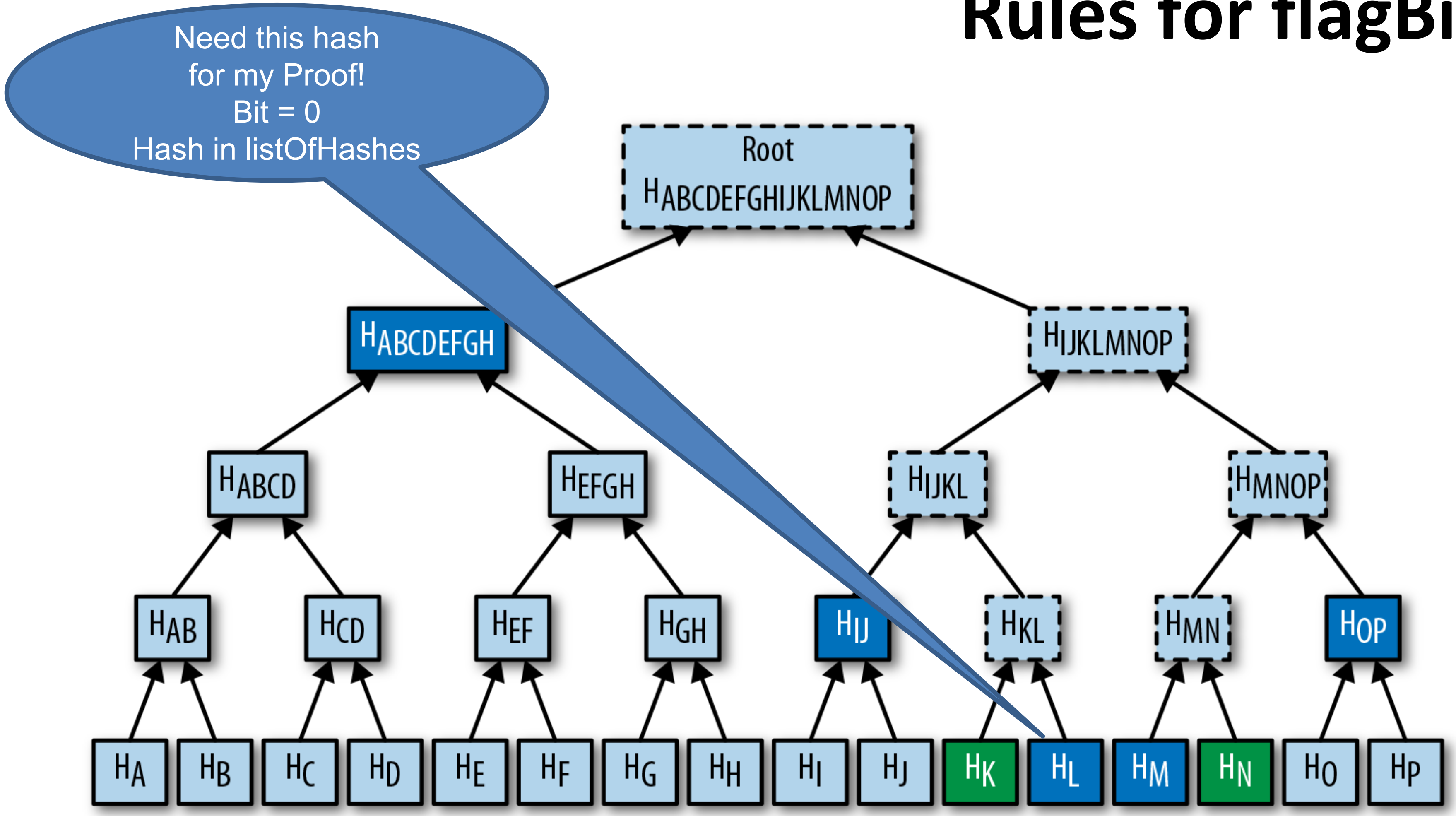
Rules for flagBits



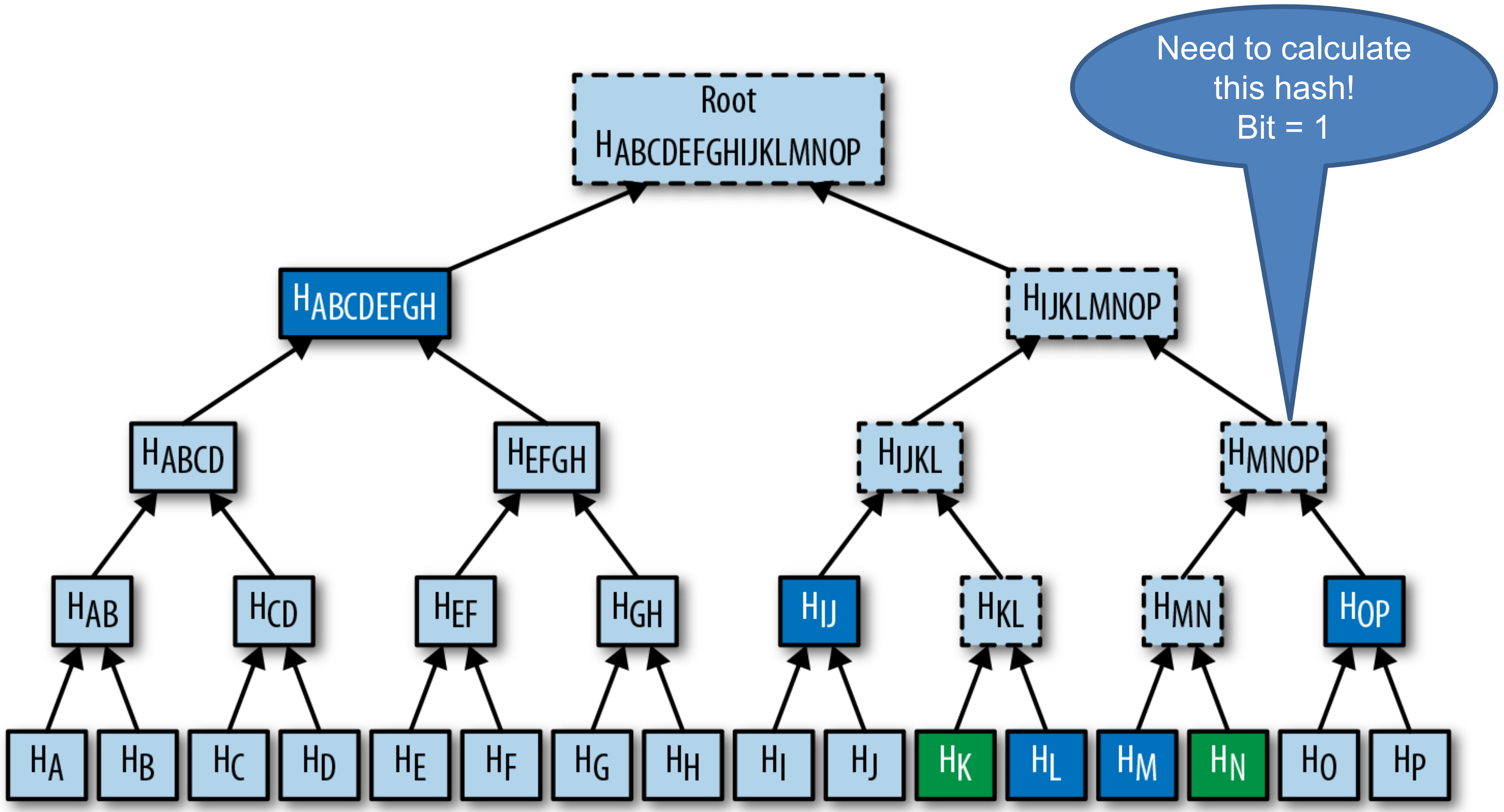
Rules for flagBits



Rules for flagBits



Rules for flagBits



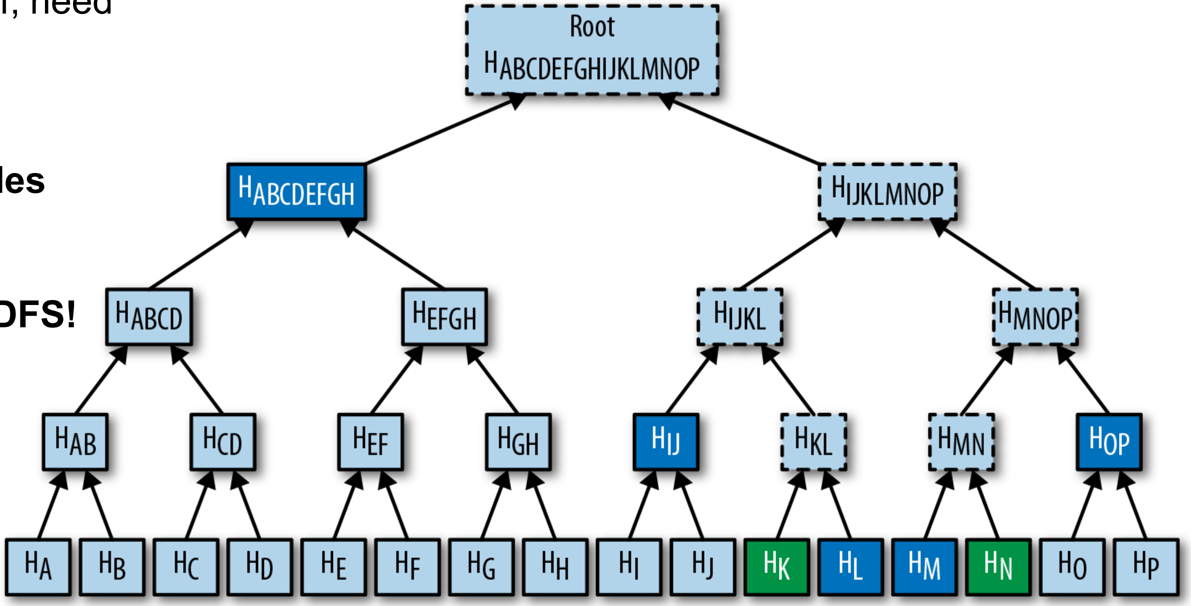
Rules for flagBits

flagBits tell me:

- Using DFS (node, left, right)
- If value is 0 no need to go down this subtree
- If value is 1 and Ip0pm not a leaf; need to parse the subtree
- If I0m a leaf I need the hash

- **flagBits consider only the nodes needed to generate the Proof!**

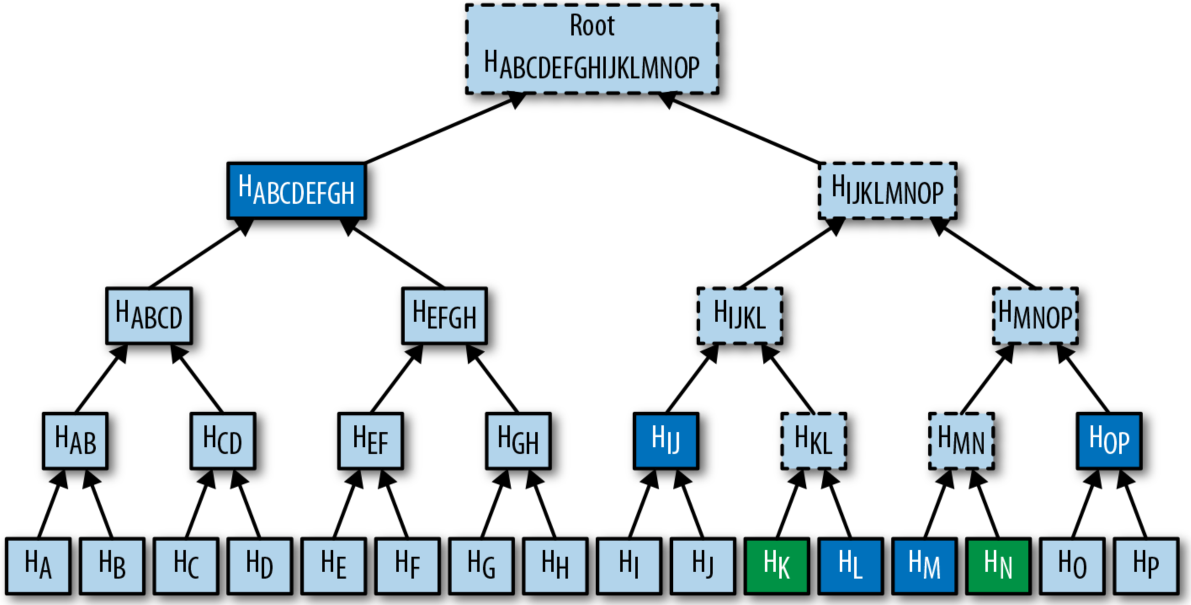
- **And the rest is processed by DFS!**



Rules for flagBits

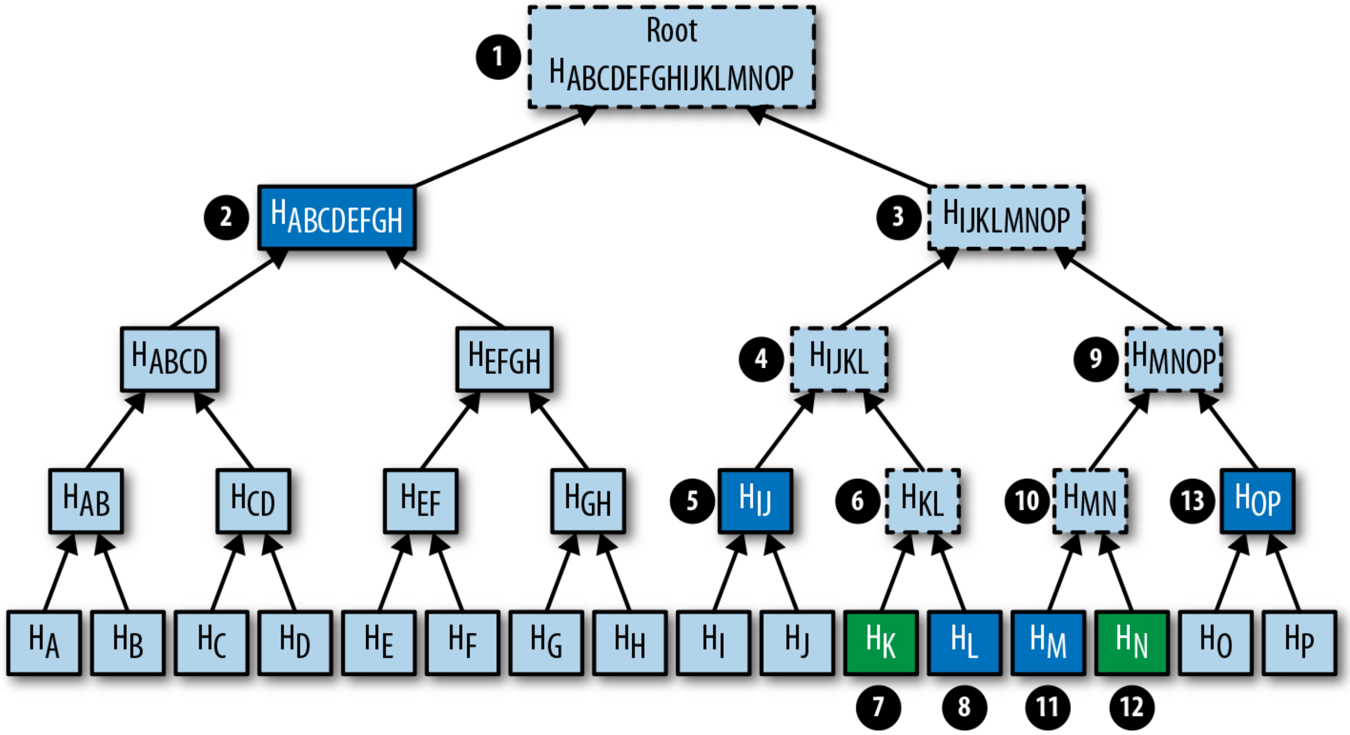
flagBits for this tree:

- 1 – root
- 0 – H(ABCDEFGH)
- 1 – H(IJKLMNOP)
- 1 – H(IJKL)
- 0 – H(IJ)
- 1 – H(KL)
- 1 – H(K)
- 0 – H(L)
- 1 – H(MNOP)
- 1 – H(MN)
- 0 – H(M)
- 1 – H(N)
- 0 – H(OP)



How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

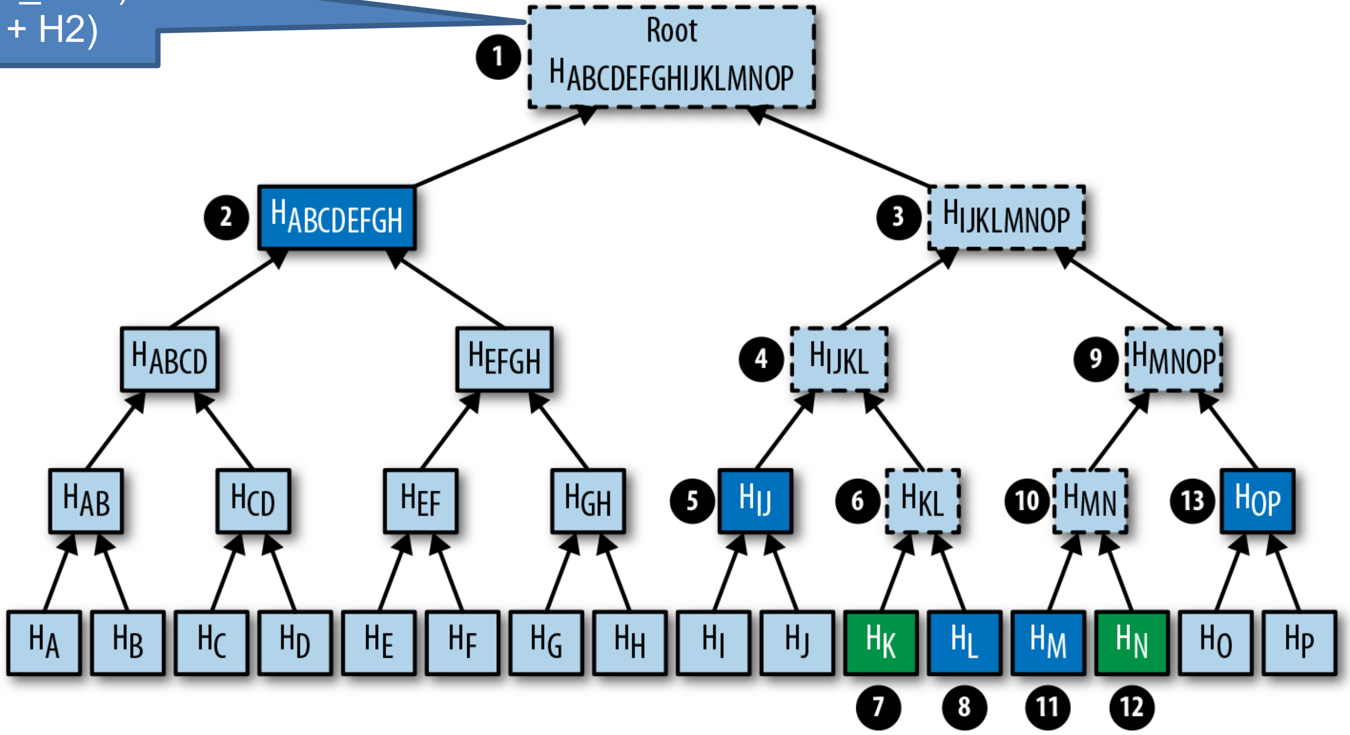


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

```
Bit = 1
H1 = DFS(left_child)
H2 = DFS(right_child)
return H(H1 + H2)
```

Proof = (nrLeaves, listOfHashes, flagBits)

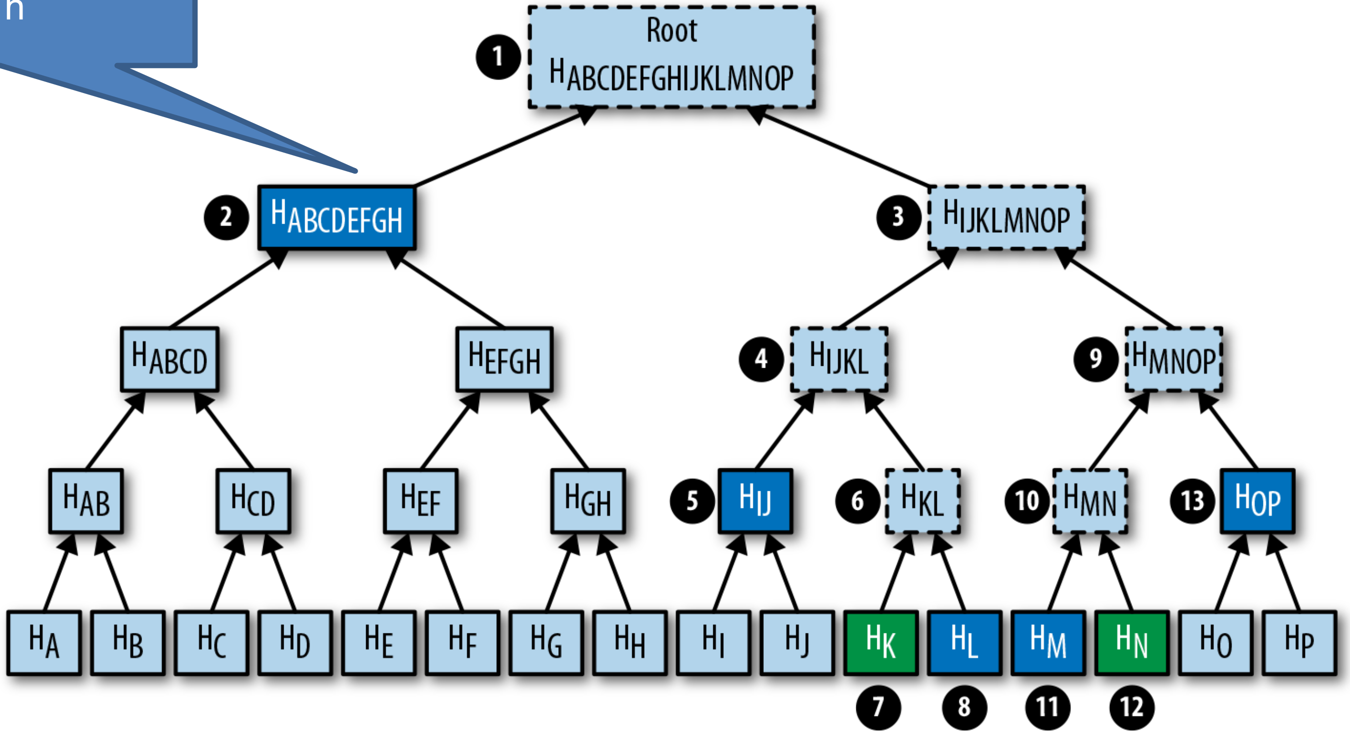


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Bit = 0
h = listOfHashes.pop(0)
return h

Proof = (nrLeaves, listOfHashes, flagBits)

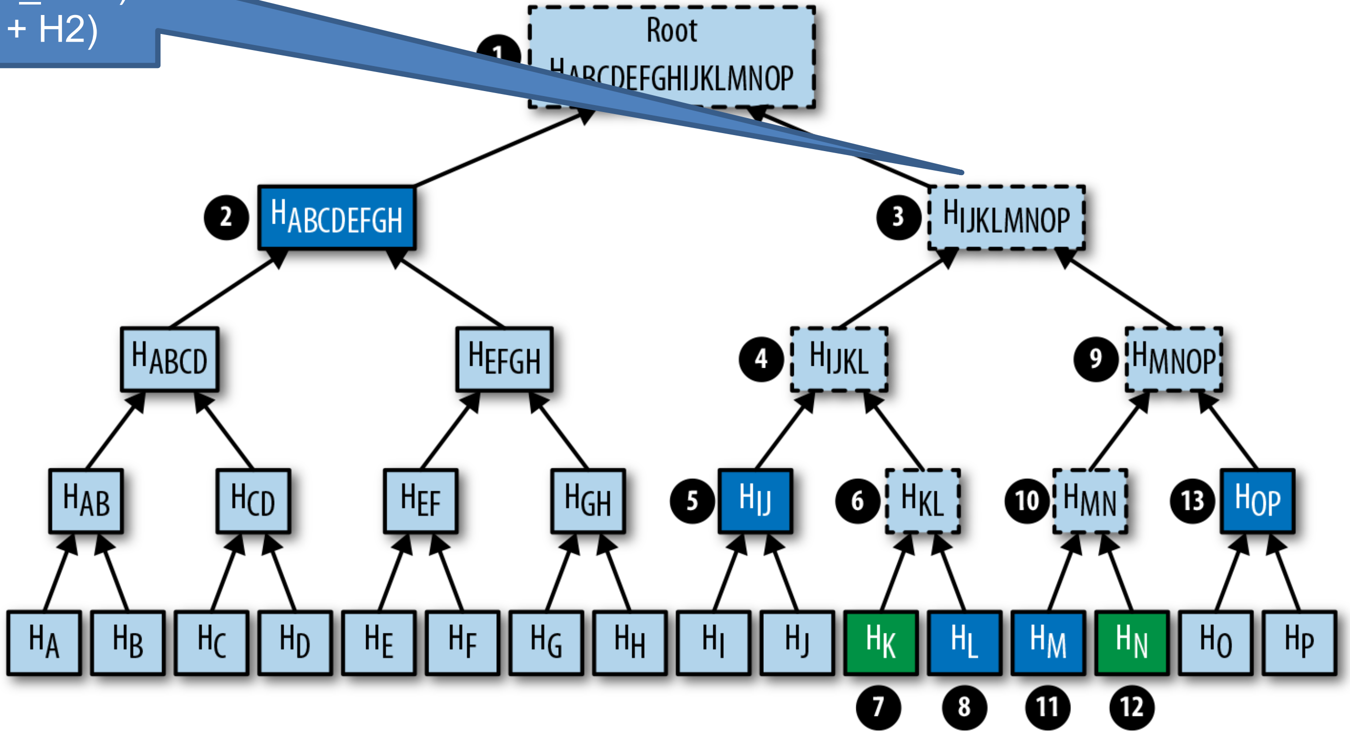


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Bit = 1
H1 = DFS(left_child)
H2 = DFS(right_child)
return H(H1 + H2)

Proof = (nrLeaves, listOfHashes, flagBits)

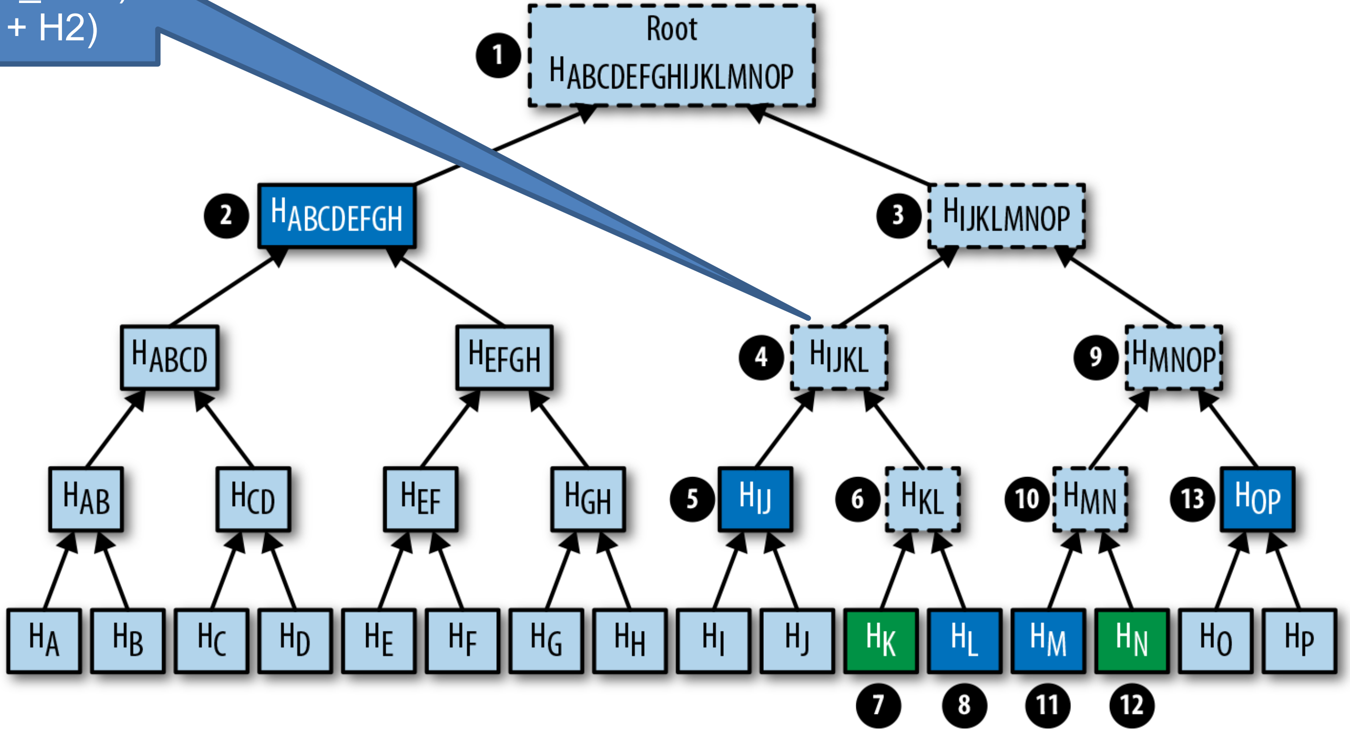


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

```
Bit = 1
H1 = DFS(left_child)
H2 = DFS(right_child)
return H(H1 + H2)
```

Proof = (nrLeaves, listOfHashes, flagBits)

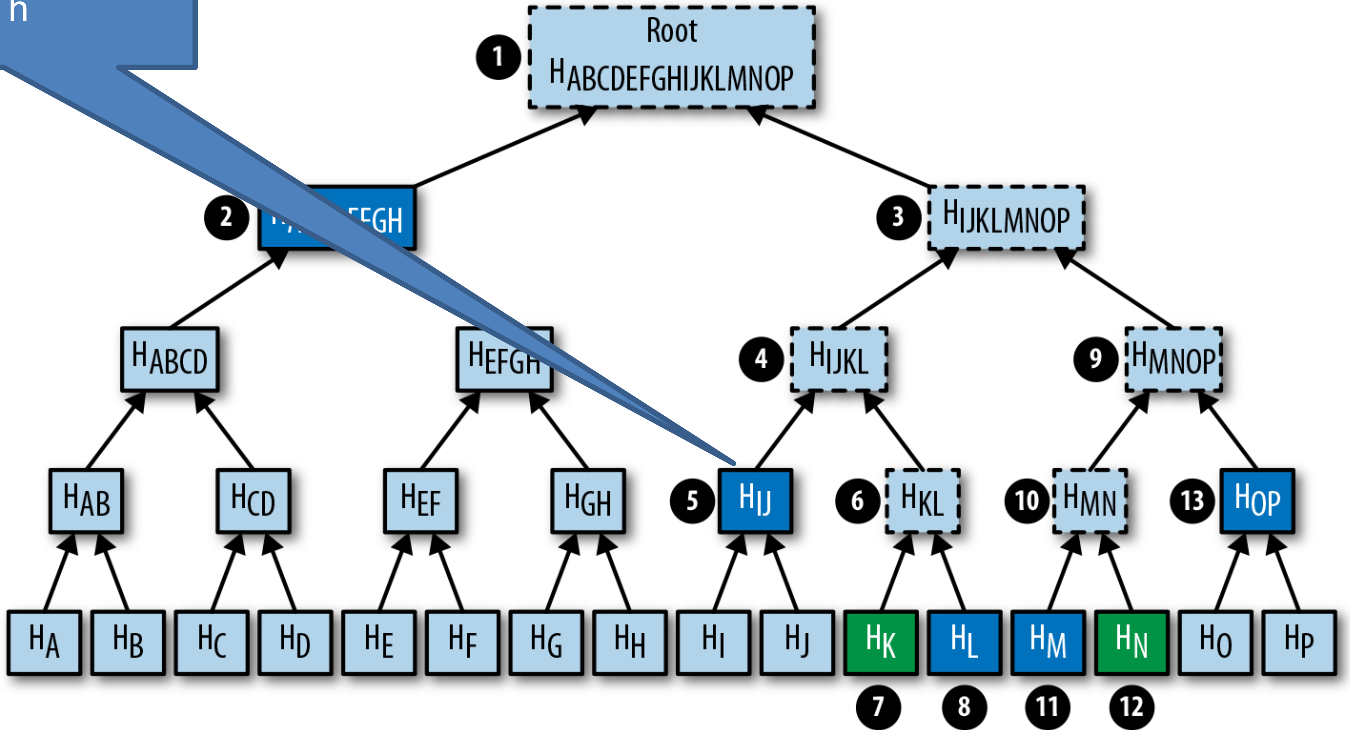


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Bit = 0
h = listOfHashes.pop(0)
return h

Proof = (nrLeaves, listOfHashes, flagBits)

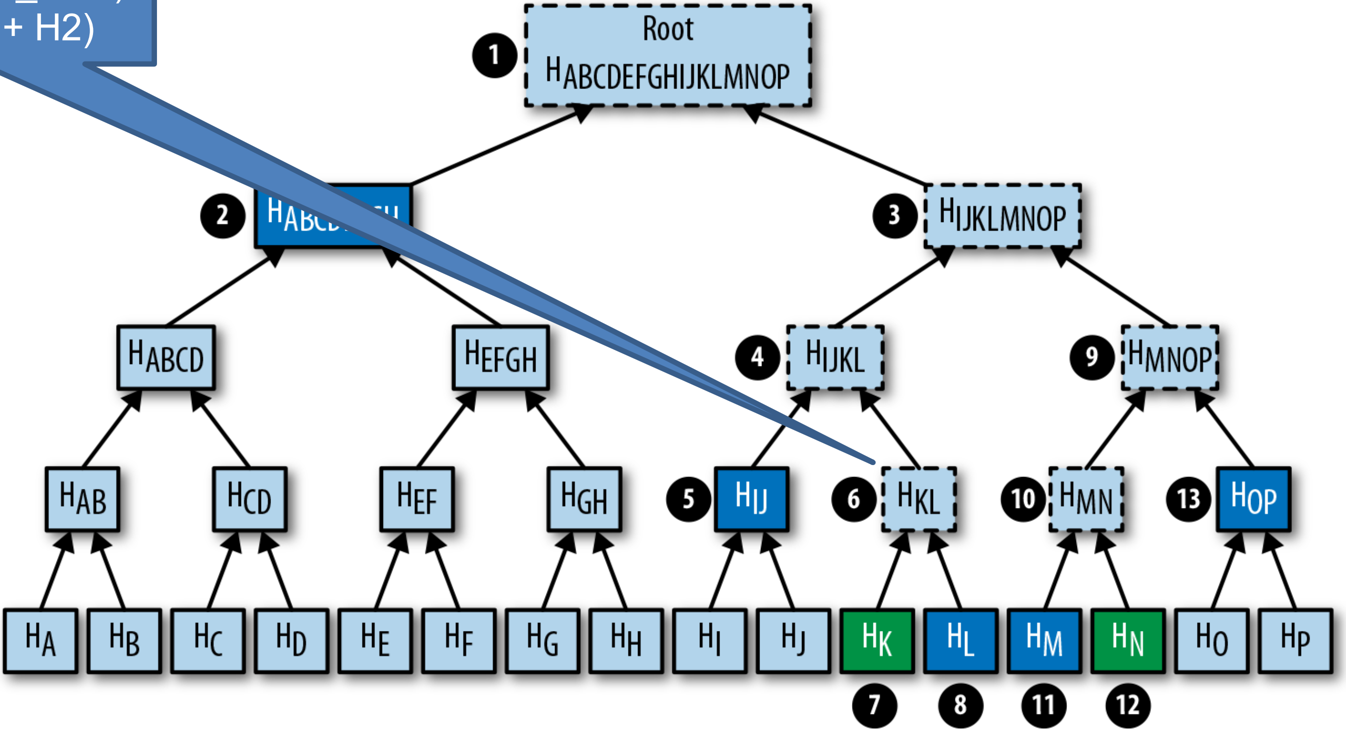


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

```
Bit = 1
H1 = DFS(left_child)
H2 = DFS(right_child)
return H(H1 + H2)
```

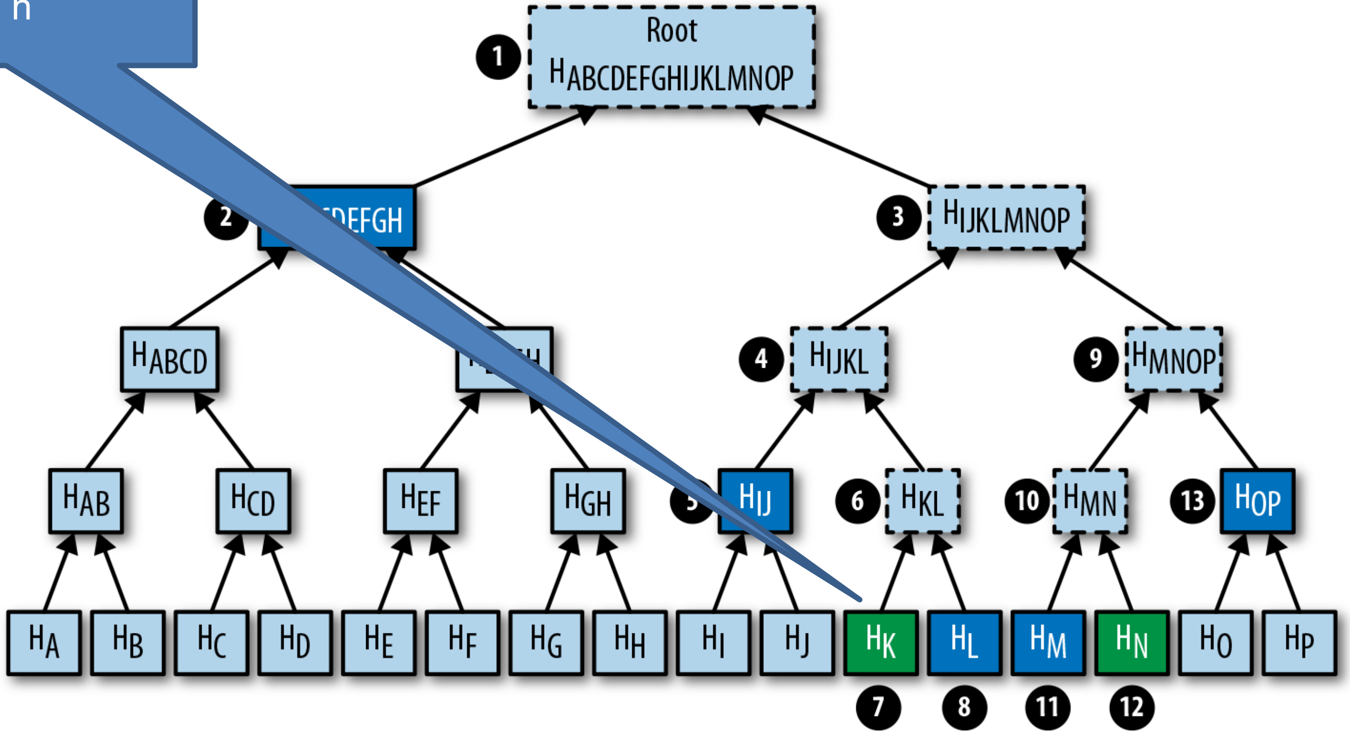


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

Bit = 1 and is a leaf
h = listOfHashes.pop0)
return h

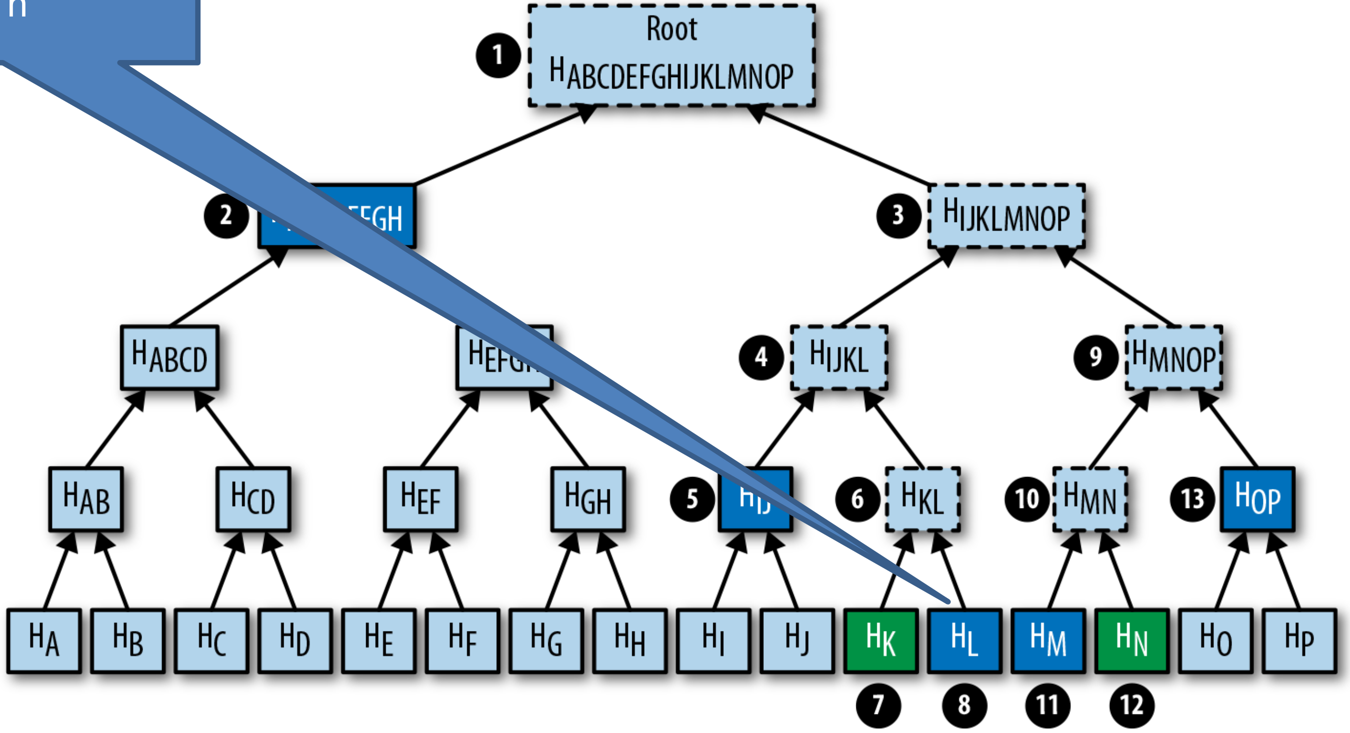


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Bit = 0
h = listOfHashes.pop(0)
return h

Proof = (nrLeaves, listOfHashes, flagBits)

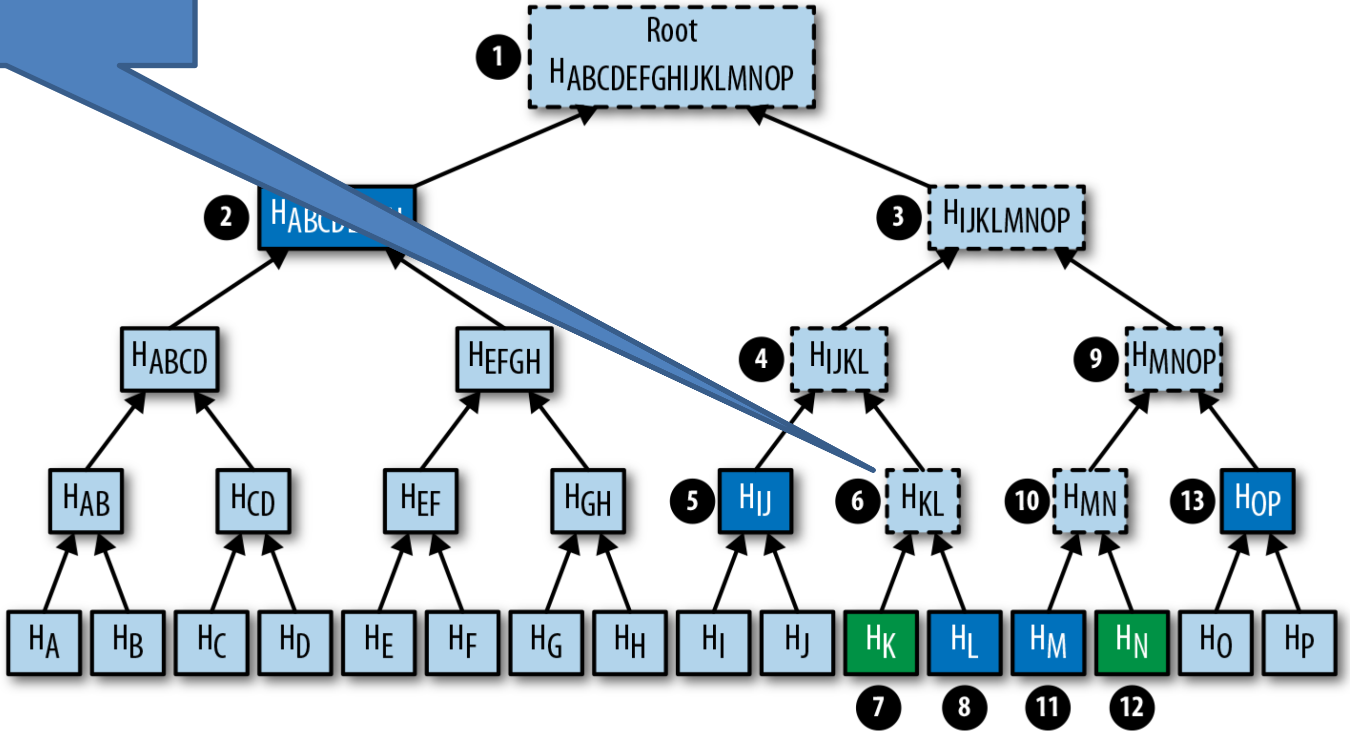


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

The node can now compute the hash!

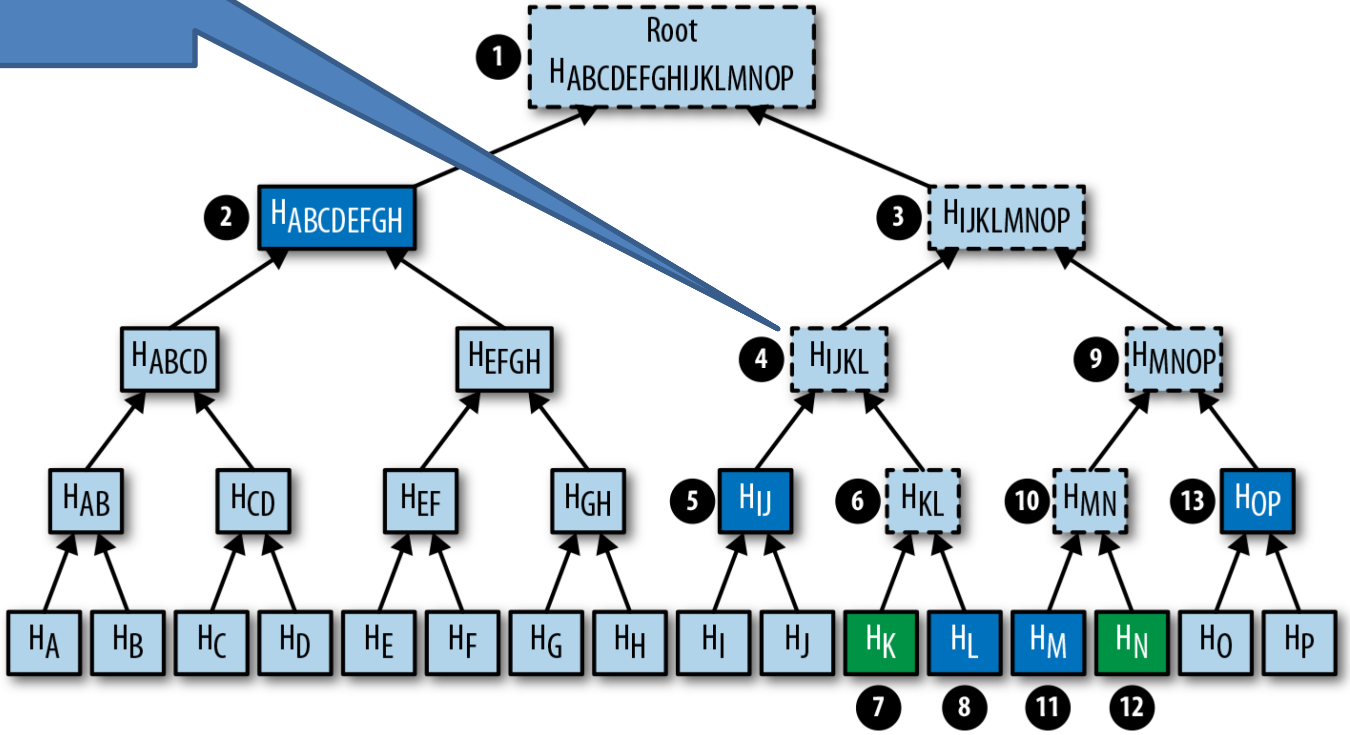


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

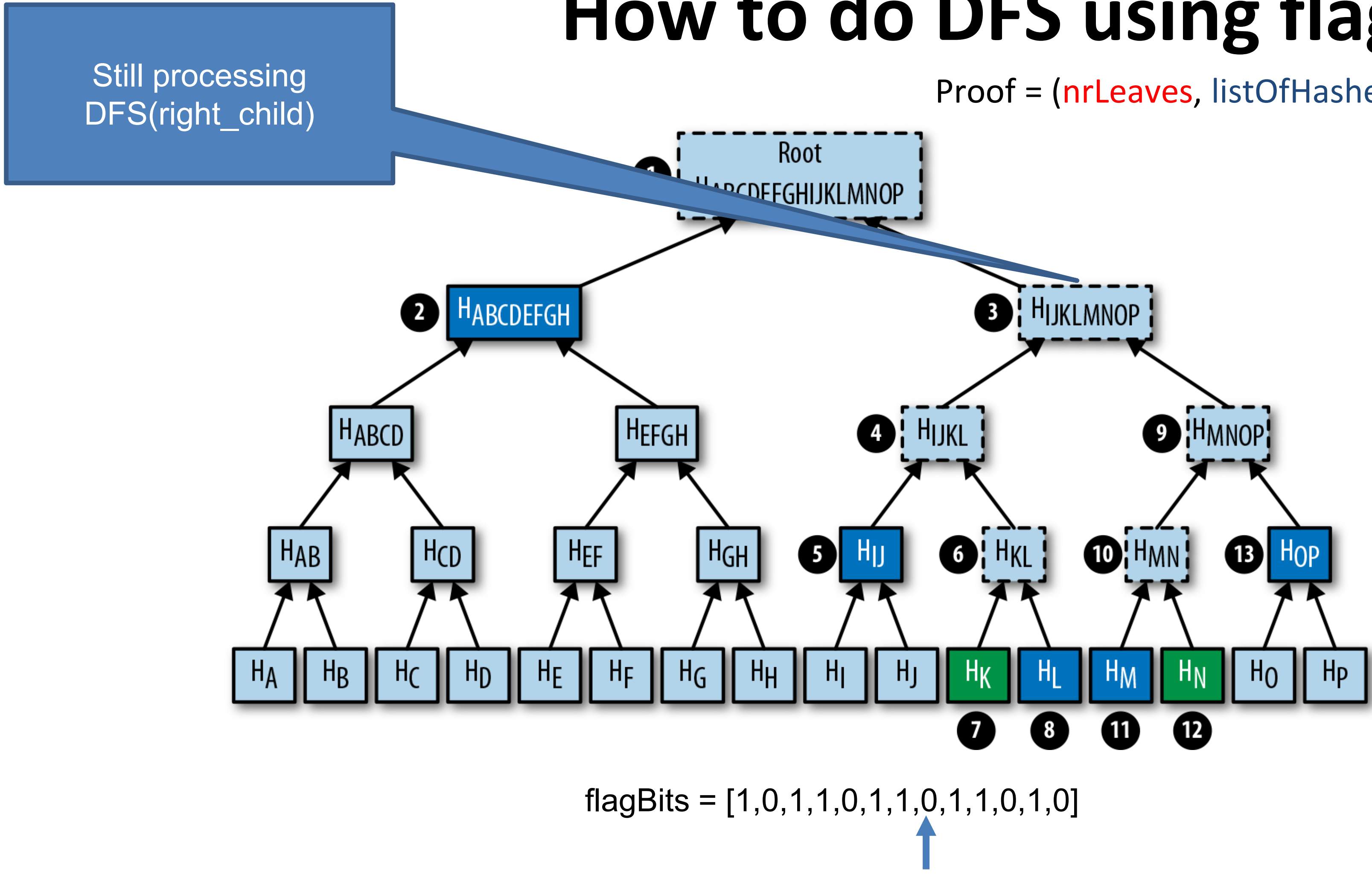
The node can now compute the hash!



flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

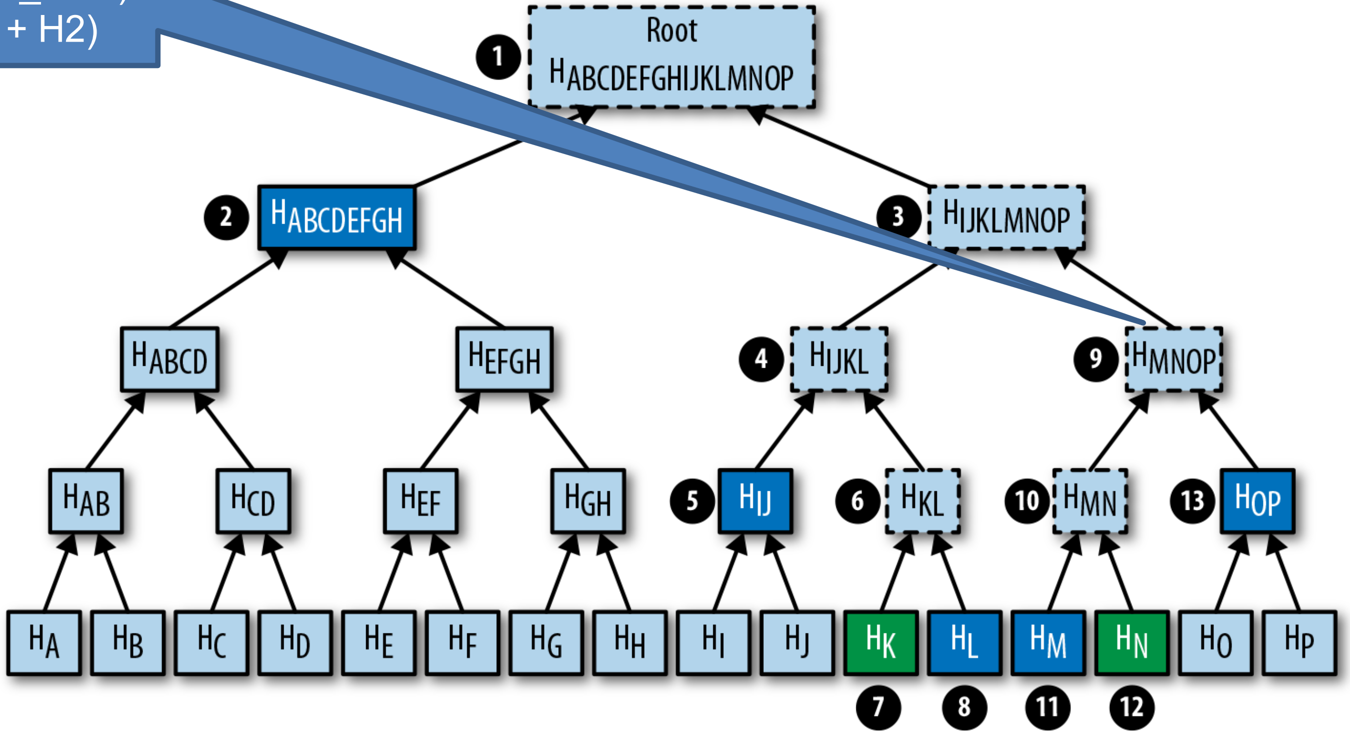


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

```
Bit = 1
H1 = DFS(left_child)
H2 = DFS(right_child)
return H(H1 + H2)
```

Proof = (nrLeaves, listOfHashes, flagBits)

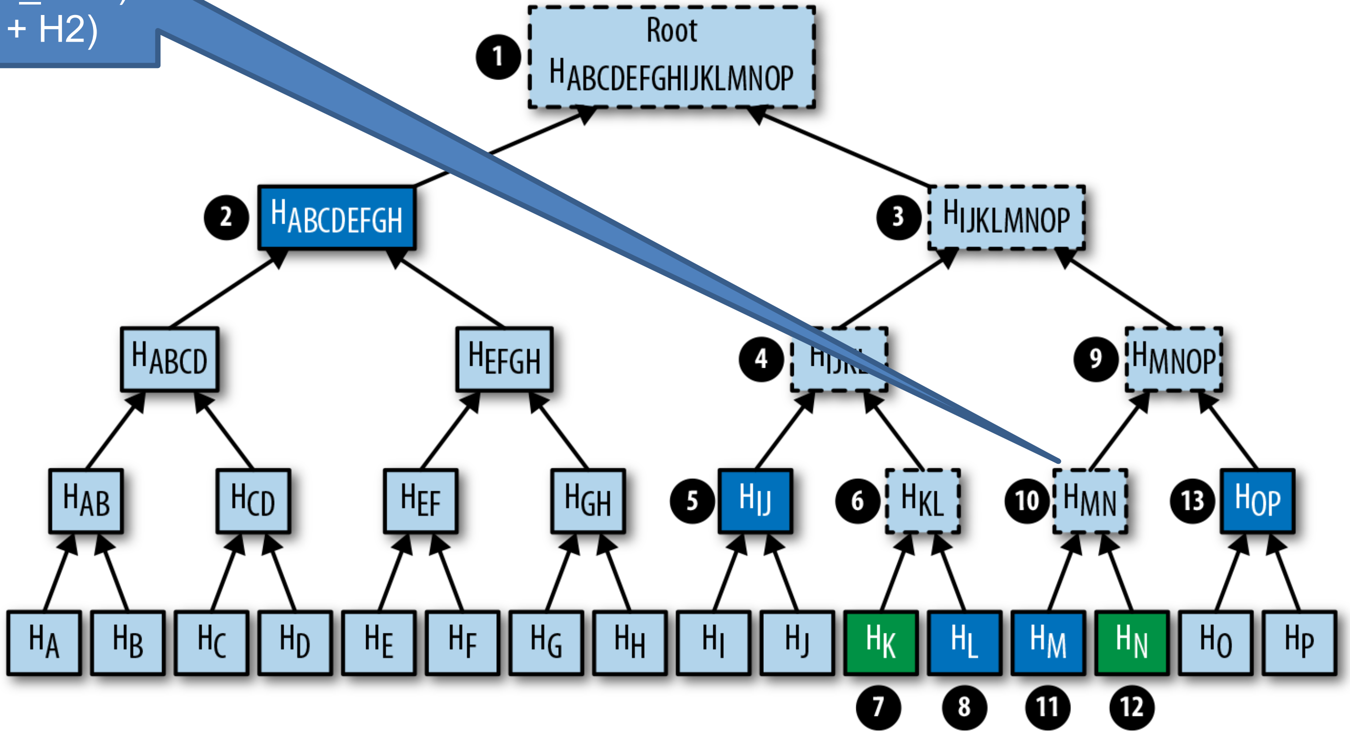


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Bit = 1
H1 = DFS(left_child)
H2 = DFS(right_child)
return H(H1 + H2)

Proof = (nrLeaves, listOfHashes, flagBits)

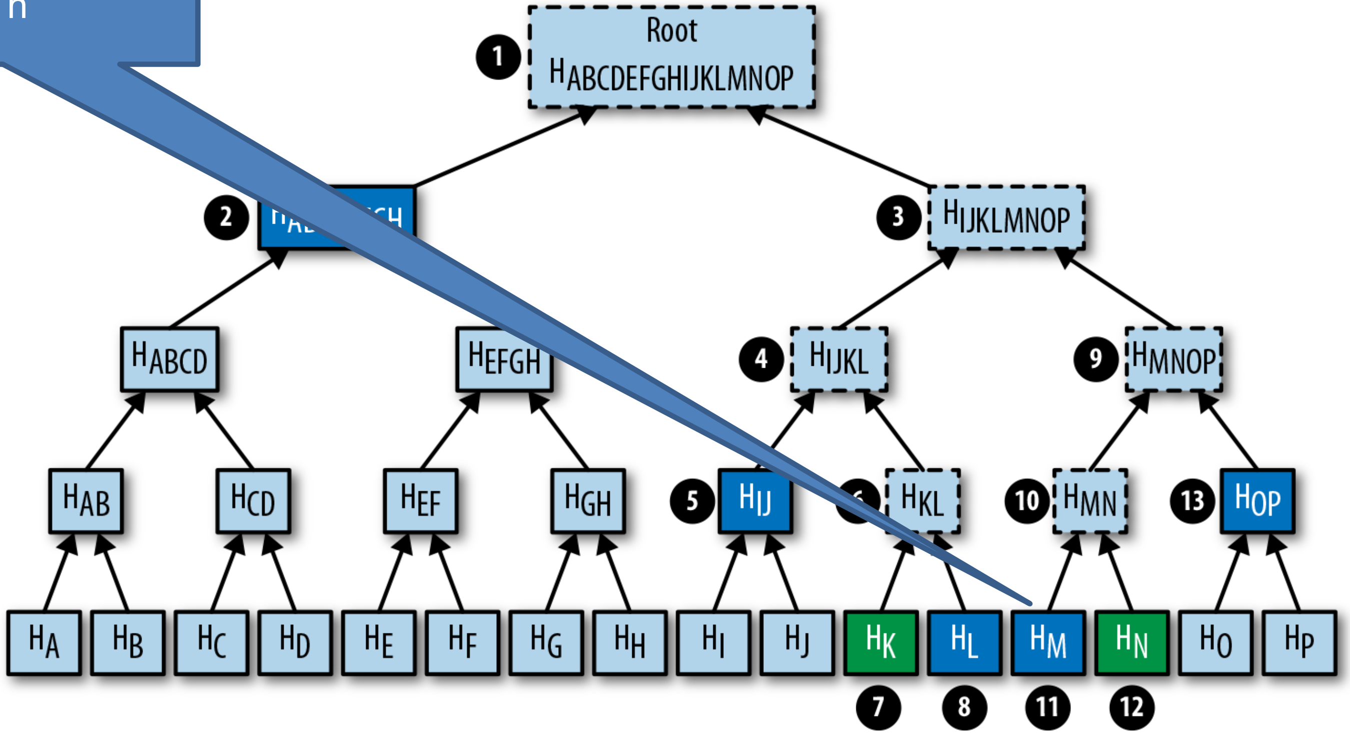


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

Bit = 0
h = listOfHashes.pop(0)
return h

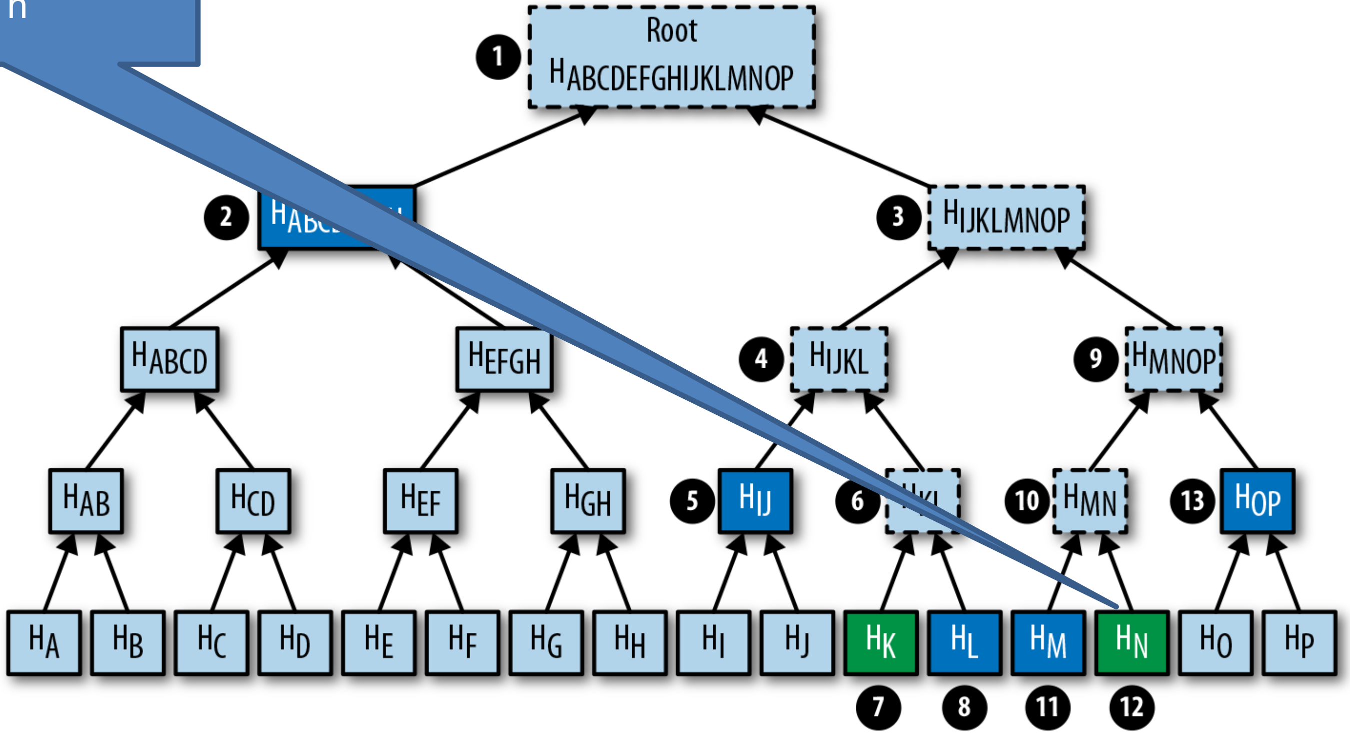


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Bit = 1 and is a leaf
h = listOfHashes.pop(0)
return h

Proof = (nrLeaves, listOfHashes, flagBits)

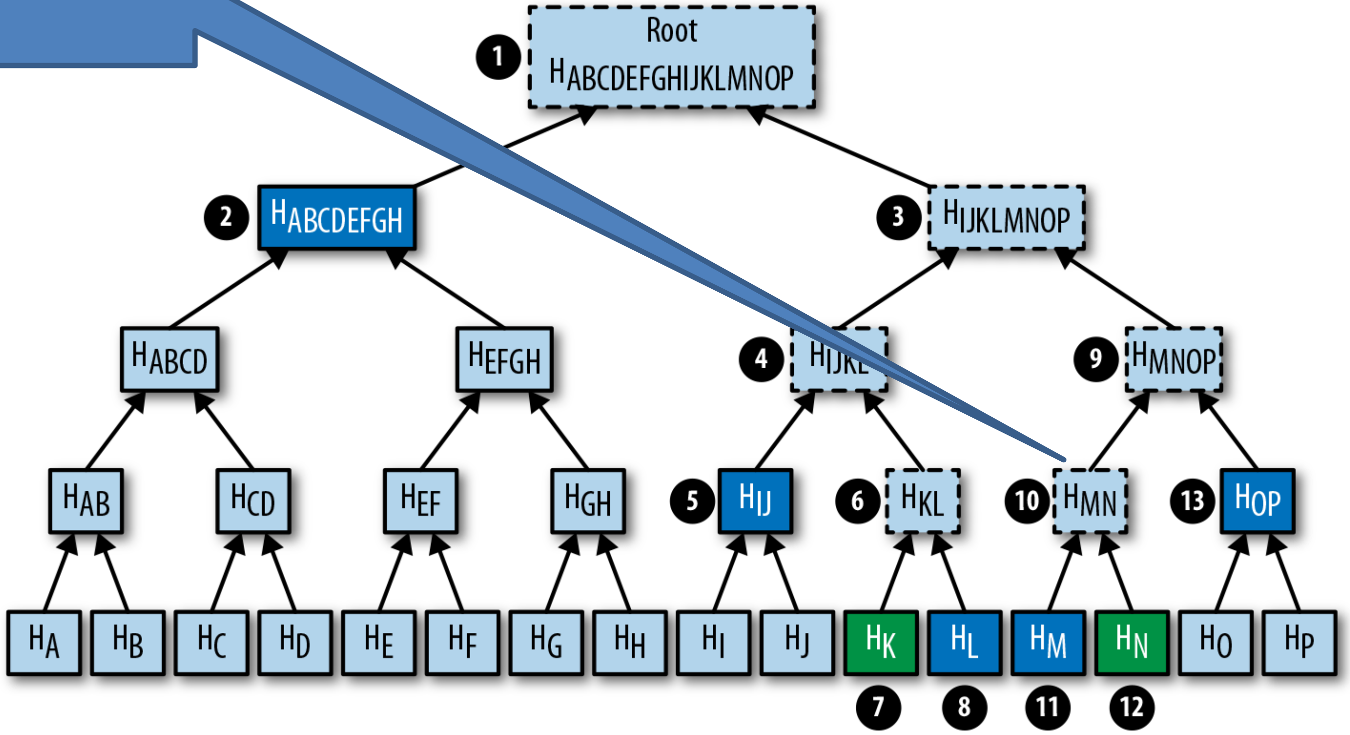


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

The node can now compute the hash!

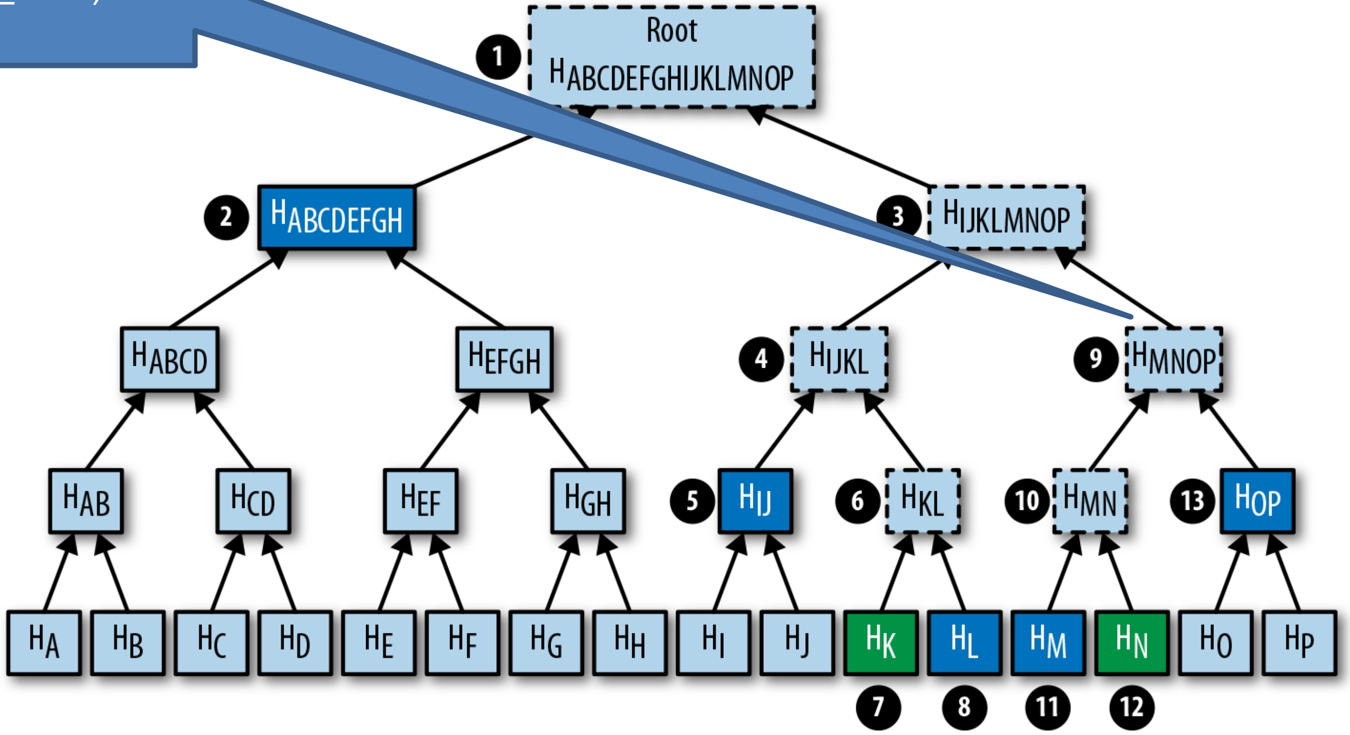


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

Still processing DFS(right_child)

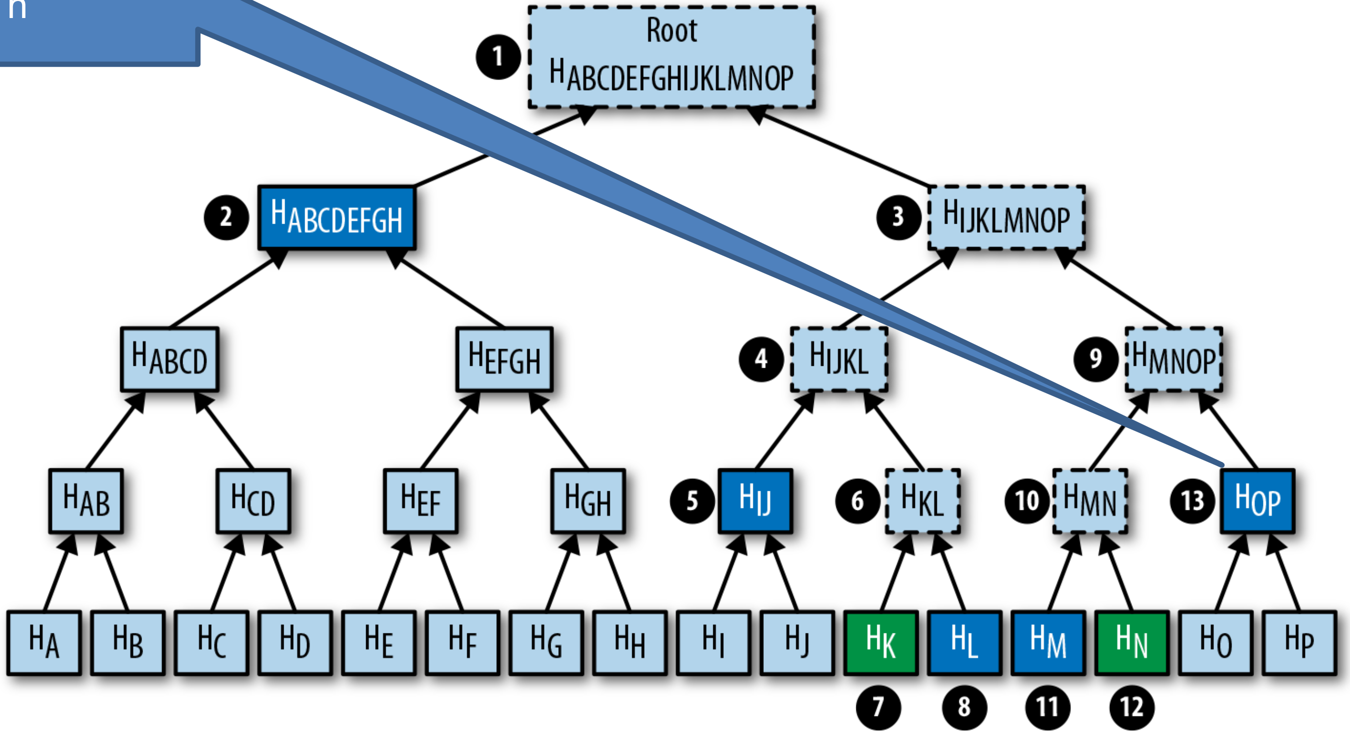


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

```
Bit = 1
h = listOfHashes.pop(0)
return h
```

Proof = (nrLeaves, listOfHashes, flagBits)

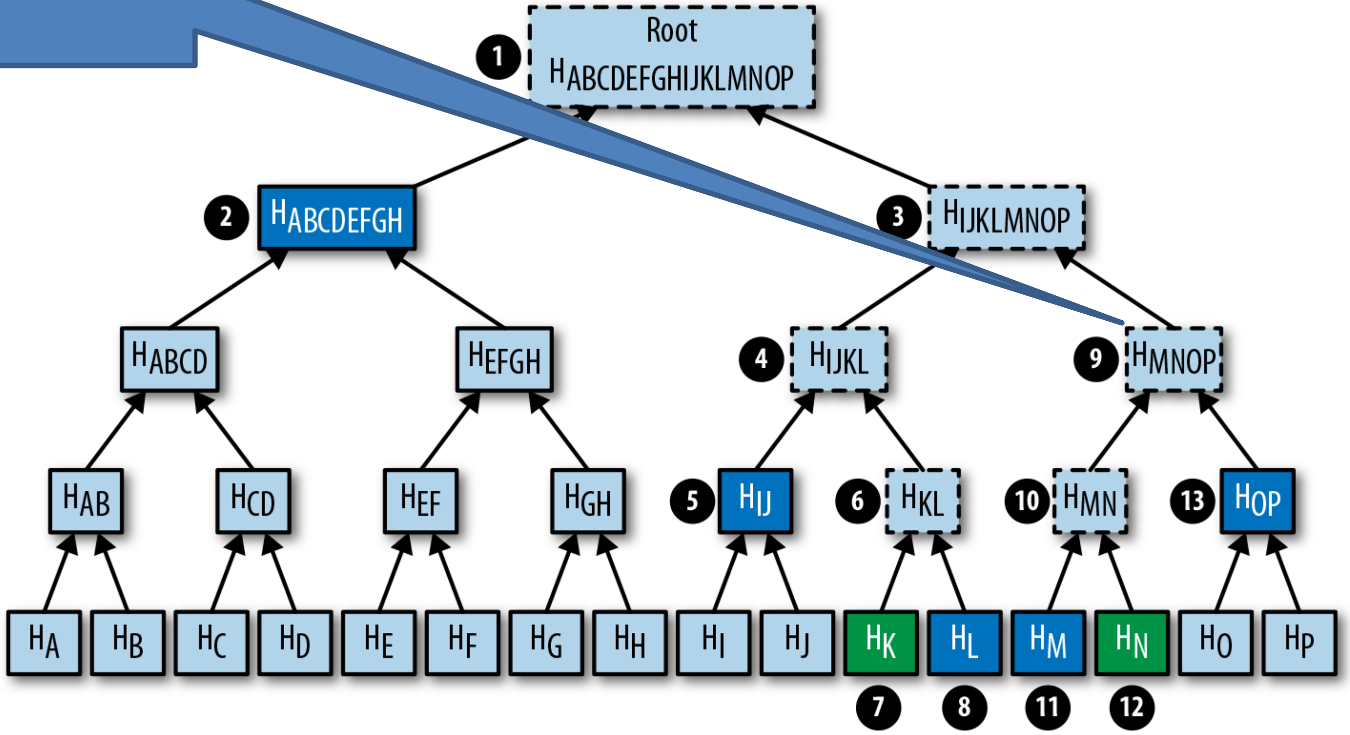


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

The node can now compute the hash!

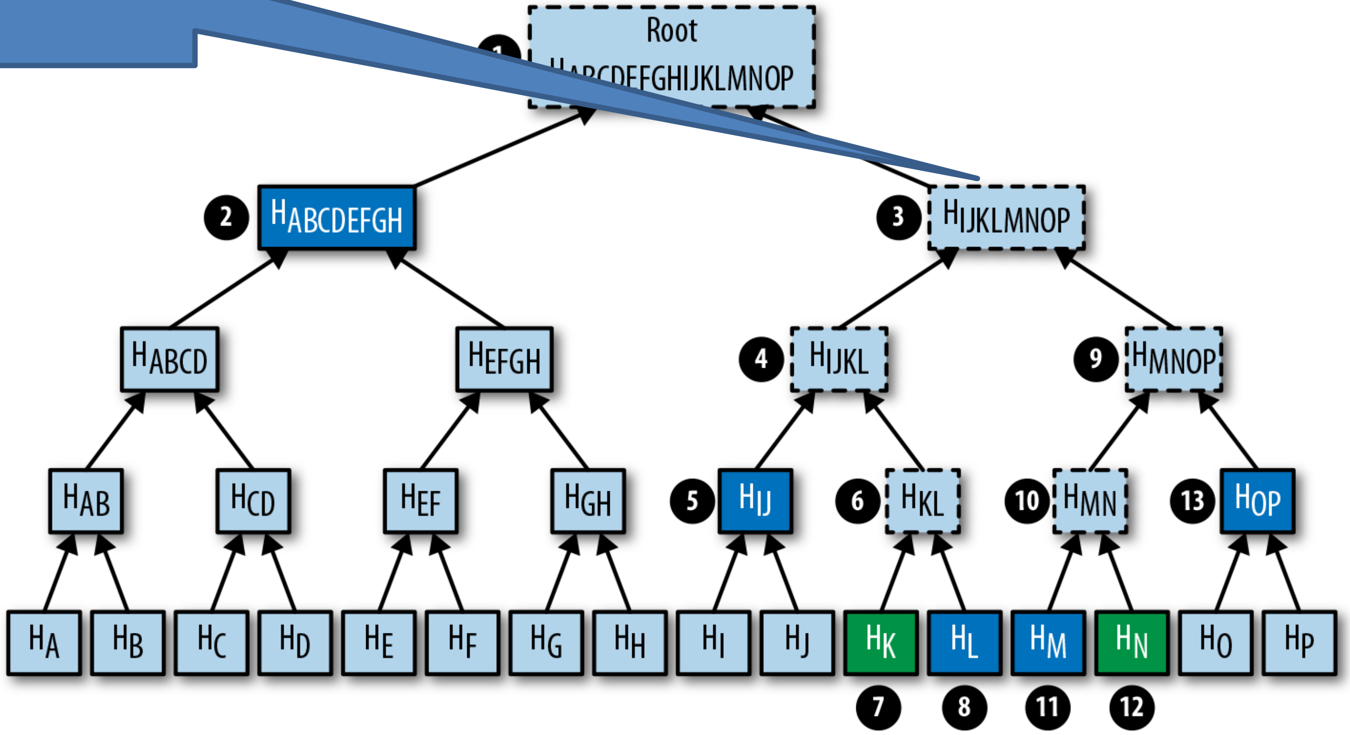


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

The node can now compute the hash!

Proof = (nrLeaves, listOfHashes, flagBits)

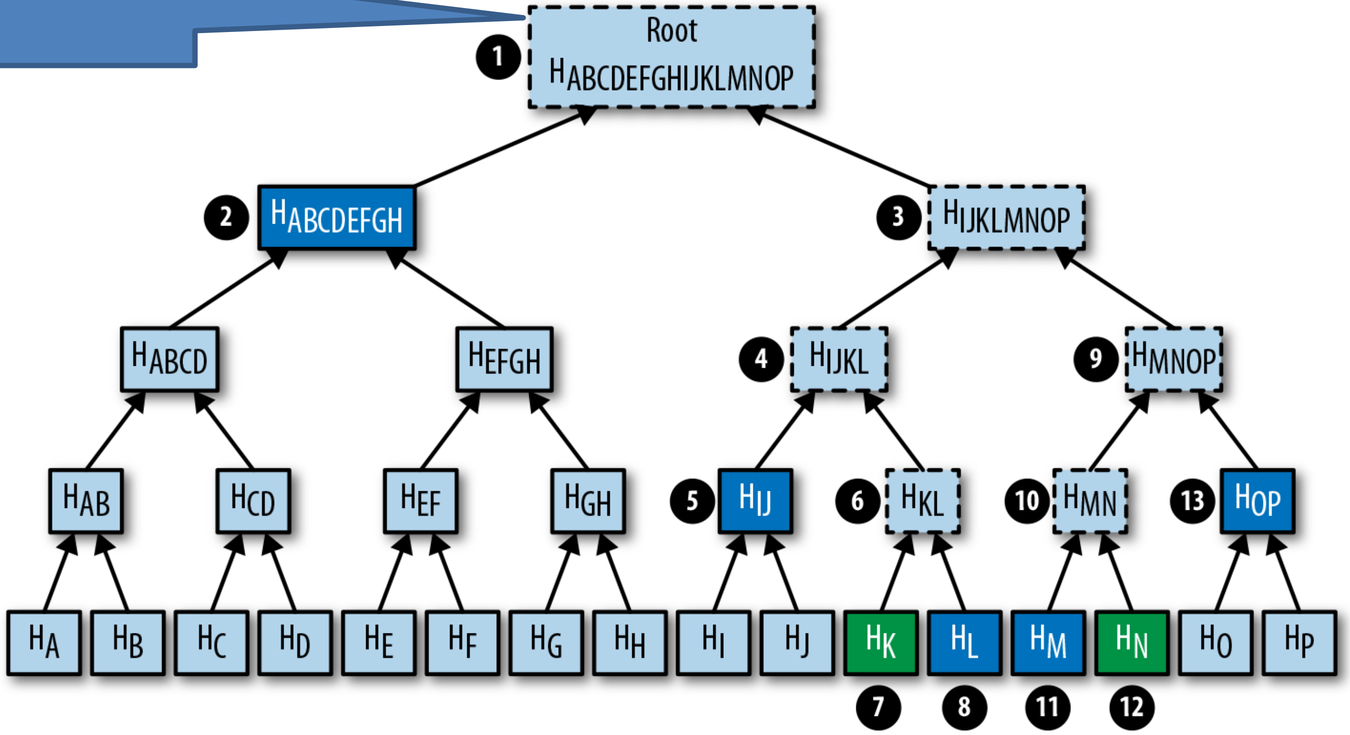


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)

We can compute the root

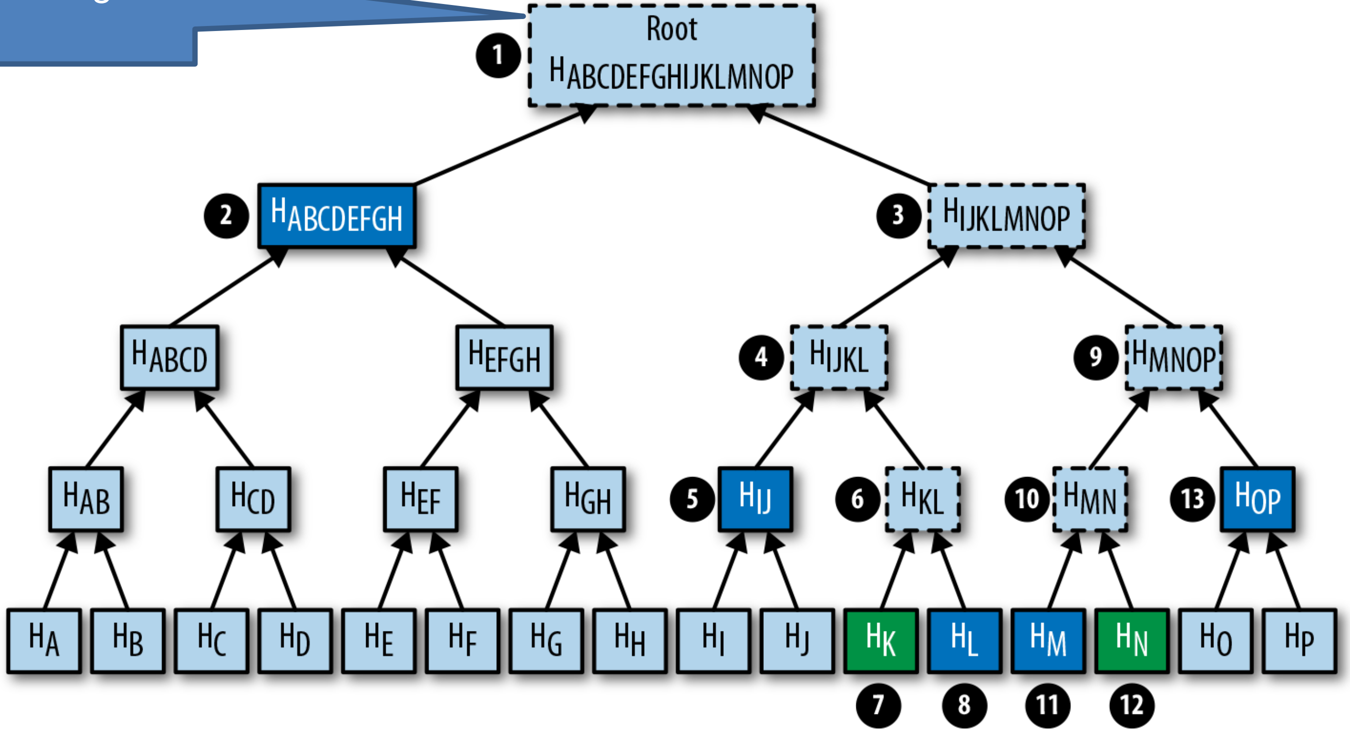


flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

If the hash is equal to the one we have all good

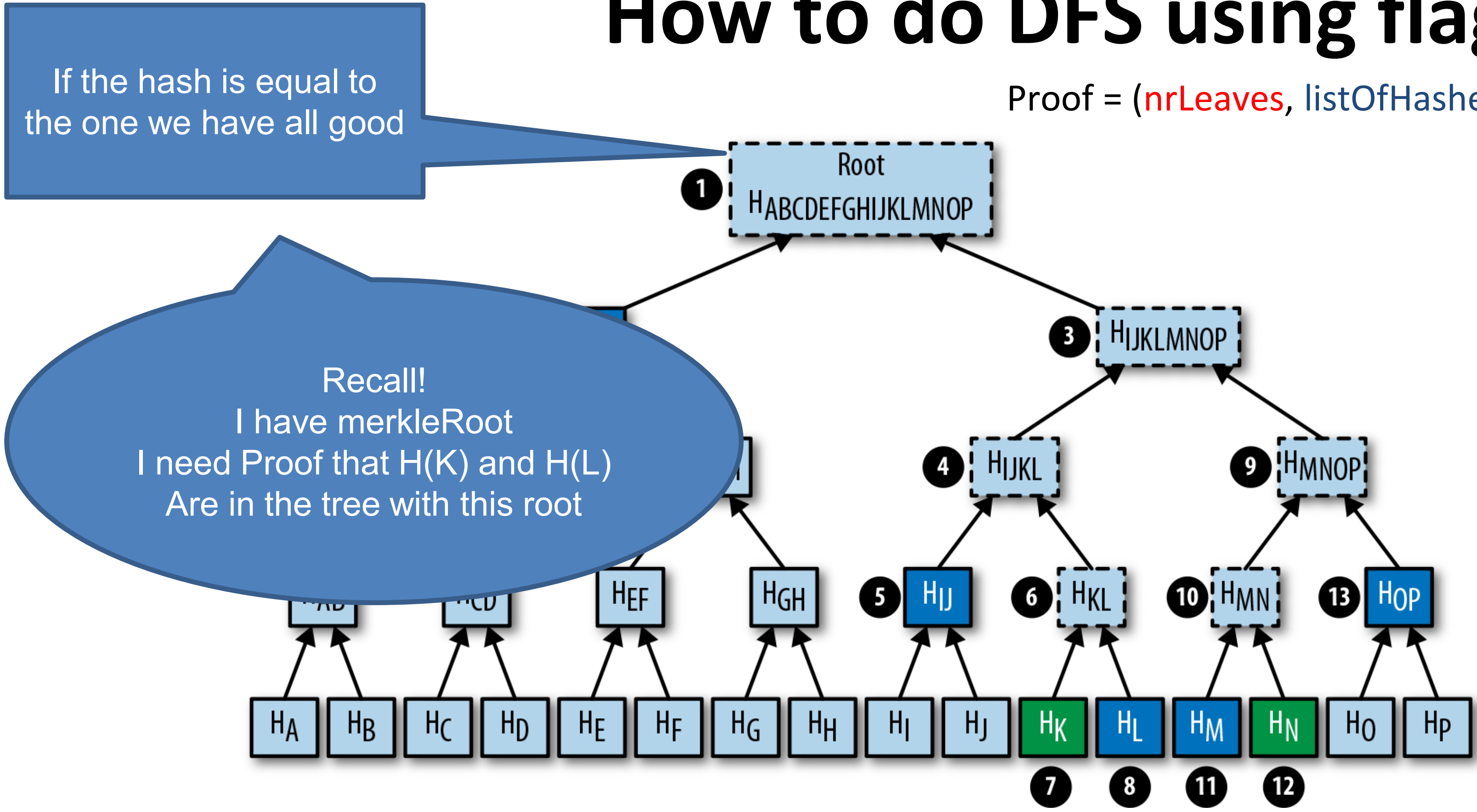
Proof = (nrLeaves, listOfHashes, flagBits)



flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

How to do DFS using flagBits?

Proof = (nrLeaves, listOfHashes, flagBits)


If the hash is equal to the one we have all good
Recall! I have merkleRoot I need Proof that H(K) and H(L) Are in the tree with this root
1 Root $H_{ABCDEFGHIJKLMNOP}$
3 $H_{IJKLMNOP}$
4 H_{IJKL}
9 H_{MNOP}
H_{AB}
H_{CD}
H_{EF}
H_{GH}
5 H_{IJ}
6 H_{KL}
10 H_{MN}
13 H_{OP}
H_A H_B H_C H_D H_E H_F H_G H_H H_I H_J H_K H_L H_M H_N H_O H_P
7 8 11 12

flagBits = [1,0,1,1,0,1,1,0,1,1,0,1,0]

Practice time!

- Now we need to implement all this!

References

- Jimmy Song, Programming Bitcoin, chapter 11