Merkle Trees : Assignment 3

One of the major security and validity checks that blockchains do is using Merkle trees.

In this assignment you will implement a Merkle tree hash.

When you pull code from git you should have a `./A-03/hash` directory. This is a copy of Assignment 2's `./A-02/hash`. The new code that you will be working on is in `./merkle`.

Pseudo Code

1. Create a slice to hold the hashes of the leaves. Each leaf hash is a `[]byte`. So make the data type `[][]byte`. Make this slice of slice of byte then length of the data. That would be `len(data)`. Let's call this `hTmp`.
2. For each data block
    1. Calculate a hash for the data block using `hash.HashOf()`.
    2. Save this in the slice created in (1) above.
3. Create a `[][]byte` slice to hold the intermediate hashes in the tree. This will need to be no more than `len(data)/2+1` in length. The plus 1 is so that 0 blocks of hasing or an odd number of blocks will have enough space. Let's call this `hMid`.
4. Start with `ln = len(data)/2+1`
5. While `ln >= 1`
    1. For each pair of hashes (if you have an odd number just use the single hash)
        - Calculate the hash of the pair using `hash.Keccak256`. It takes a variable number of arguments so you can pass 1 or 2 arguments to it.
        - Append this to `hMid`.
    2. Replace hTmp with hMid
    3. Recalculate `ln = len(hTmp)/2`
    4. Generate a new empty hMId of length ln
6. Return `hTmp[0]`

Submit

1. Your code, ./merkle/merkle.go.
2. Any additional test cases that you created.
3. Your prove that this works.

References

1. Wikipedia has a nice discussion
2. Another explanation of Merkle Trees - with more details