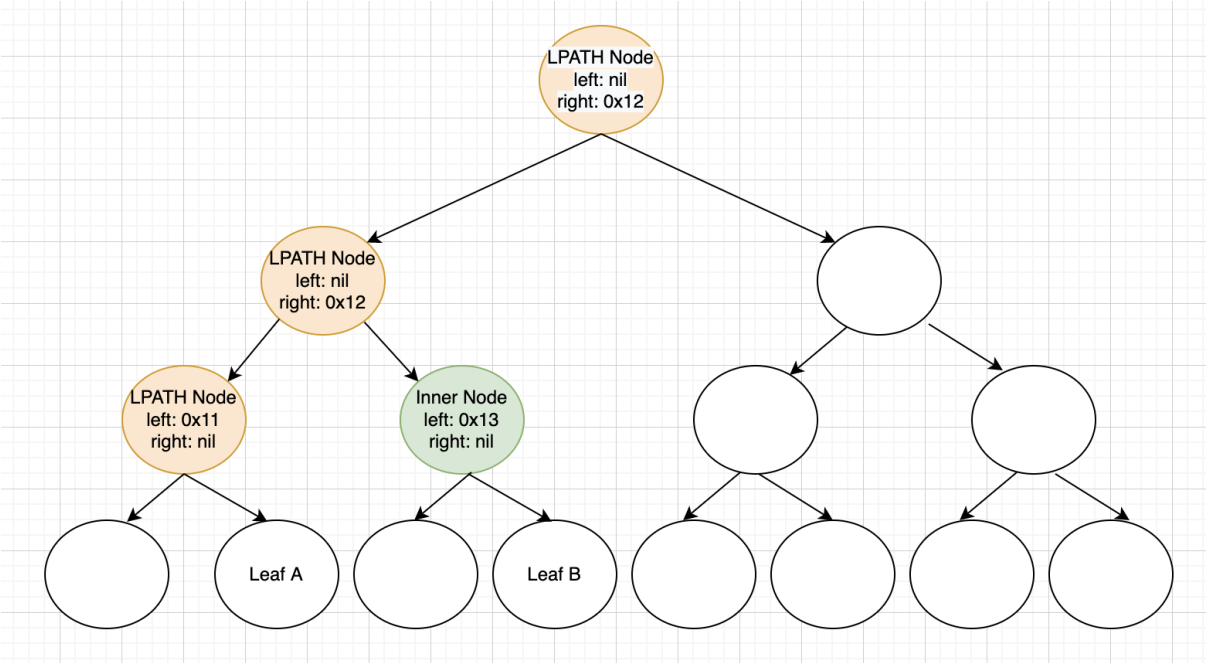
代码分支：https://github.com/guagualvcha/bsc/tree/demo_test

What is range proof:



We can prove the existence of **[leafA, leafB….leafx]** in a batch.
Leaf A is the most left leaf node, its proof is named as LPath.
For the rest leaf nodes, it always has a joined node with Lpath, for example Leaf node B only needs one proof inner node, it only requires that subtree hash equals to the right hash of the joined LPATH node.x`

Everything will be fine if both left and right are calculated within the Hash func, however, when left is not nil, `childHash` will be replaced as right hash, while we did not check the whether right hash is empty or not.

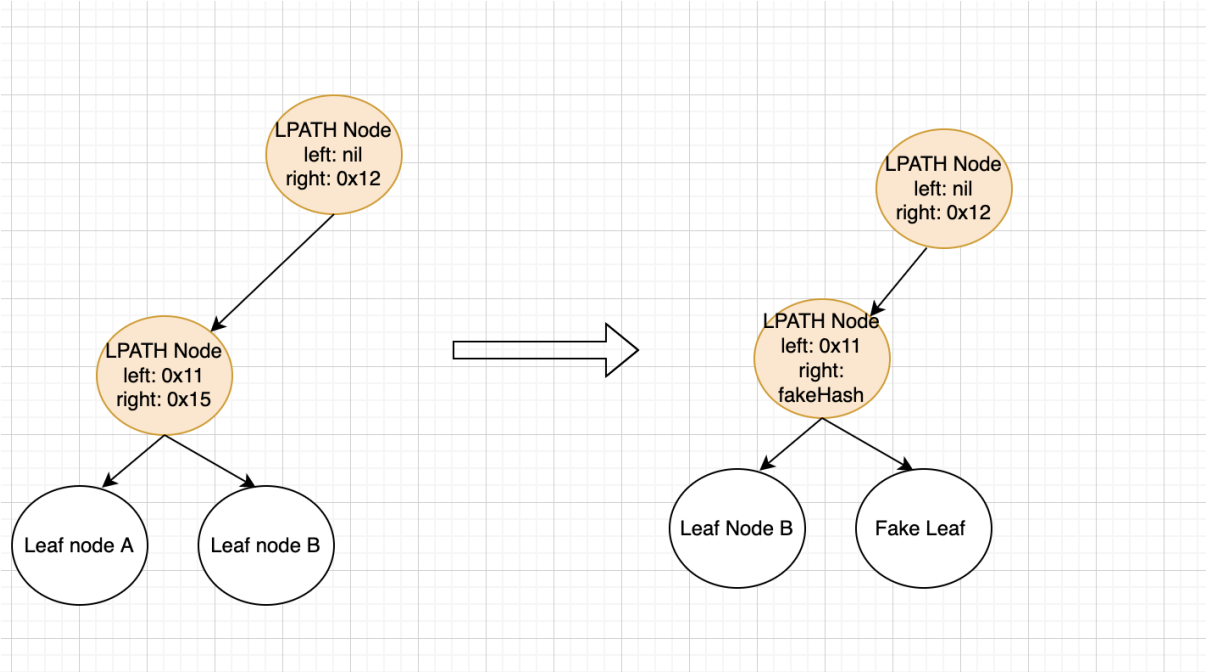
```go
func (pin proofInnerNode) Hash(childHash []byte) []byte {
    hasher := tmhash.New()
    buf := new(bytes.Buffer)

    err := amino.EncodeInt8(buf, pin.Height)
    if err == nil {
        err = amino.EncodeVarint(buf, pin.Size)
    }
    if err == nil {
        err = amino.EncodeVarint(buf, pin.Version)
    }

    if len(pin.Left) == 0 {
        if err == nil {
            err = amino.EncodeByteSlice(buf, childHash)
        }
        if err == nil {
            err = amino.EncodeByteSlice(buf, pin.Right)
        }
    } else {                        Right is not incldded
        if err == nil {
            err = amino.EncodeByteSlice(buf, pin.Left)
        }
        if err == nil {
            err = amino.EncodeByteSlice(buf, childHash)
        }
    }
```

Therefore, hacker can forge the right hash without verification:

```go
            // Recursively verify inners against remaining leaves.
            derivedRoot, treeEnd, done, err := COMPUTEHASH(inners, rightmost && rpath.isRightmost())
            if err != nil {
                return hash: nil, treeEnd, done: false, cmn.ErrorWrap(err, format: "recursive COMPUTEHASH call")
            }
            if !bytes.Equal(derivedRoot, lpath.Right) {
                return hash: nil, treeEnd, done: false, cmn.ErrorWrap(ErrInvalidRoot, format: "intermediate root hash %
            }
            if do {0x0} nil
                return hash, treeEnd, done: true, err: nil
            }
```



Verify the key value just research the leaves, and we can find leaves inside the forged proof.

Potential issue:

1. Disable range proof, there is no need for range proof actually, current mechanism does not support this.
2. Disable the proof IAVL absence operator, no issues so far, but also unnecessary.
3. Block header should not revert to old ones.
4. Research ICS.

```
// XXX: This should be managed by the rootMultiStore which may want to register
// more proof ops?
func DefaultProofRuntime() (prt *merkle.ProofRuntime) {
        prt = merkle.NewProofRuntime()
        prt.RegisterOpDecoder(merkle.ProofOpSimpleValue, merkle.SimpleValueOpDecoder)
        prt.RegisterOpDecoder(iavl.ProofOpIAVLValue, iavl.IAVLValueOpDecoder)
        prt.RegisterOpDecoder(iavl.ProofOpIAVLAbsence, iavl.IAVLAbsenceOpDecoder)
        prt.RegisterOpDecoder(ProofOpMultiStore, MultiStoreProofOpDecoder)
        return
}
```

5. Check the right and left hash, they can not exit at the same time.