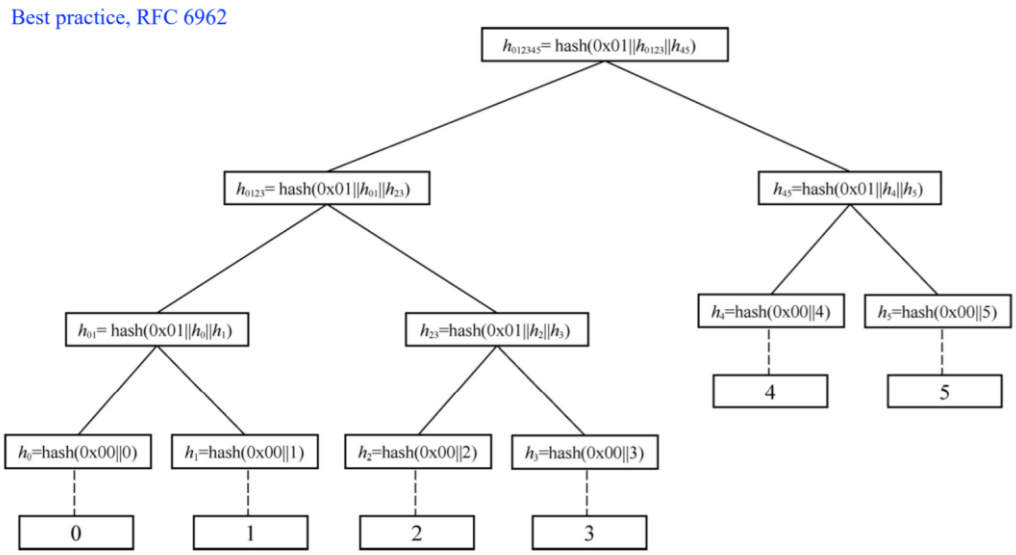
Merkle Tree



*Project: Impl Merkle Tree following RFC6962

- Construct a Merkle tree with 10w leaf nodes
- Build inclusion proof for specified element
- Build exclusion proof for specified element

代码说明：

首先构建一个节点的类结构

```
class treenode:#节点
    def __init__(self,hashd,l=None,r=None,h=0):
        self.hashd=hashd#哈希值
        self.l=l#左子节点和右子节点
        self.r=r
        self.h=h#树的高度
```

python 实现 SHA256 算法主要应用 hashlib 库,声明 hash 函数计算(0x00/0x01)||h_i||h_{i-1} 的哈希值

```
def hash(sign,string):#计算哈希值
    temp=sign+string
    out = hashlib.sha256(temp.encode("utf-8")).hexdigest()
    return out
```

声明函数 calc 将输入的数据与 0x00 级联(0X00||h_i)计算哈希值，存入一个列表中

```
def calc(lst):#将数据转换为哈希值列表
    temp=[]
    for i in range(len(lst)):
        temp.append(treenode(hash("0x00",lst[i])))
    return temp
```

声明函数 rhigh 通过判断条件 $2^{i-1}<n<2^i$ 得到树的高度 i

```
def rhigh(n):#求树的高度
    temp=1
    i=0
    while 1:
        if n<=temp:
            return i
        temp=temp*2
        i=i+1
```

因为 Merkle 树是一种完全二叉树，所以它的节点只有两种情况：两个孩子的非叶节点和无孩子的叶节点。树的最底层不一定为满，但树的倒数第二层一定为满。首先计算最底层的叶子节点，位于倒数第二层的叶子节点直接存入下一列表，递归计算父节点。

```
def tree(lst):#计算完全树
    temp=[]
    if len(lst)==1:#只有一个叶子节点时，直接将此节点作为根节点
        return lst[0]
    for i in range(0,len(lst),2):
        temp.append(treenode(hash("0x01",lst[i].hashd+lst[i+1].hashd),lst[i],lst[i
    return tree(temp)


def creattree(lst,standard,h=1):#构建树
    temp=[]
    n=len(lst)
    if len(lst)==1:#列表中只有一个节点时，此节点为根节点，并返回根节点
        return lst[0]
    for i in range(0,(2*n-2**standard),2):#计算最下一层叶子节点的父节点
        temp.append(treenode(hash("0x01",lst[i].hashd+lst[i+1].hashd),lst[i],lst[i
    for i in range((2*n-2**standard),n):#不在最下一层的叶子节点直接存入列表
        temp.append(lst[i])
    return tree(temp)
```

通过循环判断数据位置小于等于还是大于 2^n 找到数据的审计路径，并将审计路径上节点的另一方向节点哈希值和路径方向（用 0，1 区分）存入列表。将需要审计的数据与正确节点级联计算哈希值，得到计算的根节点哈希值，与正确的根节点哈希值判断此数据是否正确。

```
def path(root, x):#审计路径
    lst=[]
    temp= root
    a= root.h- 1
    while a>= 0:#高度为小于0时停止向下访问子节点
        if x<= 2**a:#判断当前数据位置在左子节点
            lst.append(temp.r.hashd)#储存在审计路径另一方向节点的hash值
            lst.append(0)#储存审计路径的方向
            temp=temp.l
            a=temp.h- 1
        else:#判断当前数据位置在右子节点
            x=x-2**a
            lst.append(temp.l.hashd)
            lst.append(1)
            temp=temp.r
            a=temp.h- 1
    return lst


def exchange(data,pdata,j):#确定数据的级联顺序
    if j==0:
        temp=data+pdata
    if j==1:
        temp=pdata+data
    return temp


def audit(root,x,data):#审计
    lst=[]
    lst=path(root,x)#得到数据的审计路径
    lst.reverse()#将列表逆序
    hashdata=hash("0x00",data)
    for i in range(0,len(lst),2):#计算根节点哈希值
        hashdata=hash("0x01",exchange(hashdata,lst[i+1],lst[i]))
    if hashdata==root.hashd:
        print("此数据包含在内")
    else:
        print("此数据不包含在内")
```

创建包含 100000 个随机数据的列表，将此列表构建 merkle 树，并分别取随机数和数据列表中的数据进行审计。

```
data=[]
for i in range(0,100000):
    data.append(str(random.randint(0,10000000000)))
l=calc(data)
standard=rhigh(len(l))
root=creattree(l,standard,1)
print(root.hashd)
x=random.randint(0,100000)
audit(root,x+1,data[x])#随机选取一个数据
audit(root,x+1,str(random.randint(0,10000000000)))#从数据列表中取出数据
```

运行指导：

直接运行

运行结果：

输出根节点的哈希值，并输出两个数据的审计结果。

```
RESTART: C:\Users\dh6\Desktop\merkle tree.py
645430ac90492bcbf30ff92cea2a6800a9c390f407c7b2dfc27dab8f416597b7
此数据不包含在内
此数据不包含在内
```