Project4 Merkle 树

一、背景概述

Merkle 树是一种高效的哈希树结构，用于验证数据集的完整性和元素成员关系。RFC6962（Certificate Transparency 标准）定义了特定的 Merkle 树构建规则，旨在通过哈希前缀区分叶子节点和内部节点，增强安全性。本文基于 SM3 哈希算法（中国密码标准，输出 256 位哈希值），构建含 10 万个叶子节点的 Merkle 树，并实现存在性证明和不存在性证明。

二、Merkle 树构建（基于 RFC6962）

1 、核心规则

RFC6962 规定：
叶子节点哈希：
leaf_hash=SM3(0x00||leaf_value)，其中 0x00 为单字节前缀（标识叶子节点），leaf_value 为原始数据。
内部节点哈希：
internal_hash=SM3(0x01||left_child||right_child)，其中 0x01 为单字节前缀（标识内部节点），left_child 和 right_child 为子节点哈希。
树为层次化二叉树，每层节点数为上一层的 ceil(m/2)（m 为上一层节点数），直至根节点。

2、 构建步骤（10 万叶子节点）

假设叶子节点原始数据为 $d_0,d_1, ...,d_{99999}$，构建流程如下：
叶子层（L0）计算
计算 10 万个叶子哈希：
$L_0[i]=SM3(0x00||d_i)$（i=0,1,...,99999）
节点数 m0=100000。
内部层逐层构建
从 L_0 开始向上迭代，每层节点由下一层相邻节点哈希拼接生成：
对于第 k 层 L_k（节点数 m_k），第 k+1 层 L_{k+1} 的节点数为 $m_k+1=ceil(m_k/2)$。

第 j 个父节点（j = 0, ..., m_{k+1}-1）的哈希为：

```
  if 2j+1 < m[k]:  # 存在右子节点
      L[k+1][j] = SM3(0x01 || L[k][2j] || L[k][2j+1])
  else:  # 仅左子节点，右子节点复用左子节点哈希
      L[k+1][j] = SM3(0x01 || L[k][2j] || L[k][2j])
```

根节点生成
逐层计算至某层节点数为 1，即为根节点。10 万叶子的树深度为 17 层（L_0 至 L_{17}），根节点在 L_{17}。

三、存在性证明

存在性证明用于验证某叶子节点 d_k（索引 k）是否在树中，核心是提供从叶子到根的路径兄弟节点哈希。

1、证明生成

初始化 current_idx=k（当前节点索引），current_level=0（当前层）。
遍历至根节点：
若 current_idx 为偶数：兄弟节点为 current_idx+1（右兄弟），记录其哈希及“右兄弟”标识。
若 current_idx 为奇数：兄弟节点为 current_idx-1（左兄弟），记录其哈希及“左兄弟”标识。
更新 current_idx=current_idx//2（父节点索引），current_level +=1。
证明内容：(k,叶子哈希 h_k,兄弟路径 proof,根哈希 root_hash)。

2、证明验证

从 h_k 开始，结合 proof 中兄弟节点哈希逐层计算父节点哈希：
若兄弟为右兄弟：
parent_hash=SM3(0x01||current_hash|| sibling_hash)。
若兄弟为左兄弟：
parent_hash=SM3(0x01||sibling_hash|| current_hash)。
最终计算结果若等于 root_hash，则证明有效。

四、不存在性证明

不存在性证明需验证元素 X 不在树中，依赖叶子节点的有序性（如按 d_i 哈希升序排列），核心是证明 X 位于两个相邻叶子之间。

1、证明生成

定位 X 的插入位置：找到最大 i 使得 $d_i<X<d_{i+1}$（$A=d_i$，$B=d_{i+1}$ 为相邻叶子）。
生成 A 和 B 的存在性证明（proof_A 和 proof_B）。
证明内容：(X,A,B,proof_A,proof_B,root_hash)。

2、证明验证

验证 A<X<B（按排序规则）。
验证 proof_A 和 proof_B 有效（计算结果等于 root_hash）。
验证 A 与 B 索引连续。
所有条件满足则 X 不在树中。

五、关键说明

安全性：前缀 0x00 和 0x01 区分叶子与内部节点，避免哈希碰撞攻击。
效率：10 万叶子的存在性证明路径长度为 17（log2(100000)≈17），证明大小仅 544 字节（17×32 字节）。
有序性：不存在性证明依赖叶子有序排列，需提前按规则排序（如原始数据哈希升序）。
通过以上步骤，可基于 SM3 和 RFC6962 构建高效、安全的 Merkle 树，并实现可靠的存在性与不存在性证明。