Merkle Patricia Tree（简称 MPT）是一种基于前缀树（Trie）结构和默克尔哈希树（Merkle Hash Tree）计算的数据结构，用于高效地存储和验证键值对集合。MPT 通常被用于区块链技术中，特别是以太坊的账户状态存储和智能合约存储。

一、我们首先介绍前缀树（Trie）结构和默克尔哈希树（Merkle Hash Tree）：
1.前缀树（Trie）结构：
MPT 使用前缀树结构作为基础数据结构，其中每个节点包含一个字符或字节，形成一个从根节点到叶节点的路径。每个节点可能有多个子节点，每个子节点与一个特定的字符或字节相关联。在 MPT 中，键值对集合中的每个键都可以通过一个唯一的字节序列进行表示，并通过在前缀树上的路径中找到相应的叶节点来检索值。

2.默克尔哈希树（Merkle Hash Tree）计算：
MPT 使用了默克尔哈希树的概念来确保数据的完整性和验证。
每个节点的值都是其子节点值的哈希结果，这样就可以对整个树形结构进行哈希计算，确保树的完整性和不可篡改性。当数据发生更改时，树中受影响的节点将通过重新计算哈希值来反映更改。

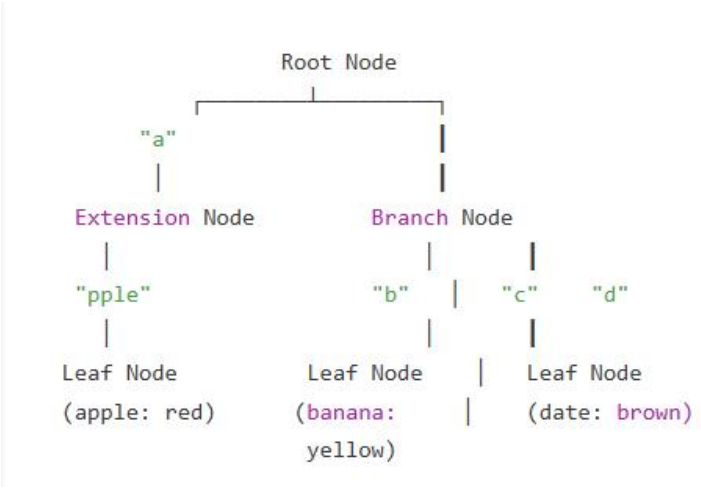
二、介绍 MPT 结构：
MPT 的结构由两个部分组成：节点和边。每个节点都有一个哈希值和相关的值。边代表节点之间的联系。
1.节点类型：
扩展节点（Extension Node）：包含一个分支路径和一个子节点的哈希值。
叶节点（Leaf Node）：包含一个键值对的哈希值。
分支节点（Branch Node）：包含 16 个槽位，每个槽位都可以连接到一个子节点的哈希值。

2.示范图例：假设我们有以下键值对集合：{"apple": "red", "banana": "yellow", "cherry": "red", "date": "brown"}
创建根节点（Root Node）：根节点是一个分支节点，初始时为空。它包含 16 个空槽位。
插入键值对：
a) 插入键值对"apple: red"：创建一个扩展节点，路径为"a"，并将其与叶节点关联。叶节点包含键值对的哈希值。
b) 插入键值对"banana: yellow"：在根节点中的第一个槽位创建一个分支节点，并将其与叶节点关联。叶节点包含键值对的哈希值。
c) 插入键值对"cherry: red"：在根节点中的第三个槽位创建一个分支节点，并将其与叶节点关联。叶节点包含键值对的哈希值。
d) 插入键值对"date: brown"：在根节点中的第四个槽位创建一个分支节点，并将其与叶节点关联。叶节点包含键值对的哈希值。

3.验证和访问值：要验证和访问键"apple"的值，我们从根节点开始沿着路径"a"向下遍历。我们首先遇到扩展节点，该节点的路径是"a"，它指向叶节点。从叶节点中获取哈希值，并使用哈希值检索存储在其他地方的键值对。在这种情况下，我们可以获取到"red"。

4.更新值：如果我们要更新键"apple"的值为"green"，我们首先需要通过遍历路径"a"找到叶节点。然后更新叶节点中存储的键值对，并重新计算叶节点的哈希值。由于叶节点的哈希值发生了变化，它的父节点（扩展节点）和祖先节点（根节点）的哈希值也需要重新计算。

如下图所示：



三、MPT 的一些性质以及应用：

1..路径压缩和编码：

MPT 对前缀树进行了优化，采用路径压缩和编码技术来减少存储空间和提高读取效率。

内部节点使用 RPL 编码（Recursive Prefix Length Encoding），将共同前缀的节点压缩为单个节点来减少存储空间。叶节点存储键值对，其中键被编码为一个唯一的字节序列。

2.持久性和不可变性：

MPT 是一种持久性数据结构，任何更改都会生成一个新的树，原始树将保持不变，这有助于历史状态的回溯和验证。由于每个节点的哈希是基于其子节点哈希计算的，所以任何更改都会导致路径上所有祖先节点的哈希值发生变化，确保了数据的不可篡改性。

3.应用：

MPT 在以太坊中被广泛应用于账户状态存储和智能合约存储。通过 MPT，可以有效地存储和检索大量的键值对，并且还可以方便地验证账户状态和智能合约的完整性。MPT 的设计使得在区块链环境中进行高效的账户状态转换和验证成为可能，同时也提供了更好的隐私性和安全性。

下附官方图解：