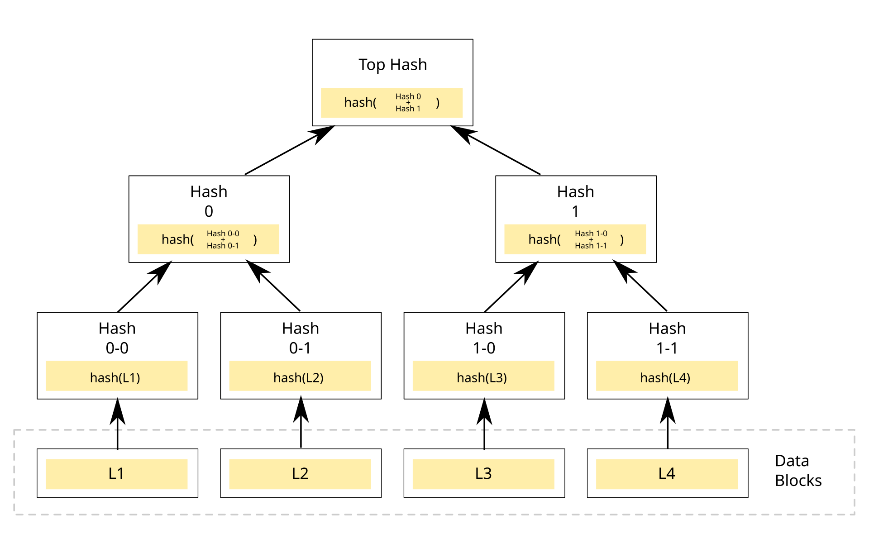
**PRACTICAL 5**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Roll No.:** | 21BCP359 |
| **Division:** | 6 | **Batch:** | G11 |
| **Aim:** | Implementation of Merkle Tree. | | |

**Merkle Tree**

A Merkle tree is a data encryption structure used in data management applications where data is sent through a hashing algorithm in different ways to create a hash that represents all of the data in a file.

|  |  |  |
| --- | --- | --- |
| **Complexity** | **Average** | **Worst** |
| **Space** | O(n) | O(n) |
| **Search** | O(log2(n)) | O(logk(n)) |
| **Insert** | O(log2(n)) | O(logk(n)) |
| **Delete** | O(log2(n)) | O(logk(n)) |



* Merkle tree also known as hash tree is a data structure used for data verification and synchronization.
* It is a tree data structure where each non-leaf node is a hash of its child nodes.
* All the leaf nodes are at the same depth and are as far left as possible.
* It maintains data integrity and uses hash functions for this purpose.

**Program**

import hashlib

class MerkleTreeNode:

    def \_\_init\_\_(self, value):

        self.left = None

        self.right = None

        self.value = value

        self.hashValue = hashlib.sha256(value.encode("utf-8")).hexdigest()

def buildTree(leaves, f):

    nodes = []

    for i in leaves:

        nodes.append(MerkleTreeNode(i))

    while len(nodes) != 1:

        temp = []

        for i in range(0, len(nodes), 2):

            node1 = nodes[i]

            if i + 1 < len(nodes):

                node2 = nodes[i + 1]

            else:

                temp.append(nodes[i])

                break

            f.write(

                "Left child : " + node1.value + " | Hash : " + node1.hashValue + " \n"

            )

            f.write(

                "Right child : " + node2.value + " | Hash : " + node2.hashValue + " \n"

            )

            concatenatedHash = node1.hashValue + node2.hashValue

            parent = MerkleTreeNode(concatenatedHash)

            parent.left = node1

            parent.right = node2

            f.write(

                "Parent(concatenation of "

                + node1.value

                + " and "

                + node2.value

                + ") : "

                + parent.value

                + " | Hash : "

                + parent.hashValue

                + " \n"

            )

            temp.append(parent)

        nodes = temp

    return nodes[0]

inputString = input("Enter the leaves as a comma-separated list (e.g., 'a,b,c,d'): ")

leaves = inputString.split(",")

with open("merkle.tree", "w") as f:

    root = buildTree(leaves, f)

**Output**

