**Merkle Tree Data Structure:**

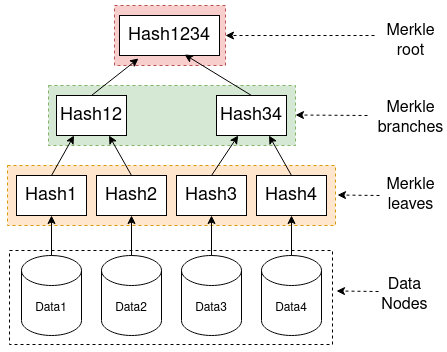
* It is known as **hash tree** which is a data structure.
* It is used for **data verification and synchronization**.
* It **maintains data integrity** and **uses hash functions** for this purpose.

**Hash function:**

* A hash function **maps an input to a fixed output** and this output is **called hash**.
* The **output is unique for every input** and this **enables fingerprinting of data**. So, huge amounts of **data can be easily identified** through their hash.

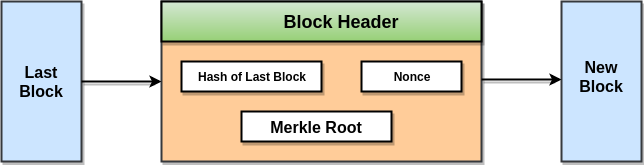
**Example: A Basic Merkle Tree**

The following figure illustrates a very small Merkle tree:



In this figure,

* The bottom nodes (***Data1*-*Data4***) are the **actual data processed** by the application.
* Each of these is **summarized by their respective hash value** (*Hash1*-*Hash4*), as a ***Merkle leaf***.
* From these, the Merkle tree **builds a hierarchy, combining hashes together until only one is left.**
* The nodes **combining other hash nodes are called *Merkle branches***(here *Hash12* and *Hash34*). When there is **only one left** (here *Hash1234*), this is called the***Merkle root*.**
* **Merkle Root is stored in the block header**. The block header is the **part of the bitcoin block** which gets hash in the process of mining. **It contains the hash of the last block, a Nonce, and the Root Hash of** all the transactions in the **current block** in a Merkle Tree.
* So having the Merkle root in block header makes the transaction tamper-proof. As this Root Hash includes the hashes of all the transactions within the block, these transactions may result in **saving the disk space**.



This is a binary merkel tree, the **top hash is a hash of the entire tree.**

* This structure of the tree **allows efficient mapping of huge data** and **small changes** made to the data **can be easily identified**.
* If we want to know where data change has occurred then we can check if data is consistent with root hash and we will not have to traverse the whole structure but only a small part of the structure.
* The **root hash is used as the fingerprint for the entire data**

*Note:* *Merkle trees are in a binary tree, so it requires an even number of leaf nodes. If there is an odd number of transactions, the last hash will be duplicated once to create an even number of leaf nodes.*

**Applications:**

* Merkle trees are **useful in** **distributed systems** where same data should exist in multiple places.
* Merkle trees can be **used to check inconsistencies**.
* **Apache Cassandra uses** Merkle trees to detect inconsistencies between replicas of entire databases.
* It is used in **bitcoin and blockchain**.