**Q1)** **A *verifier* knows the size of a Merkle tree and the root hash *r* of the tree. How can a *prover* provide a short proof that “If the *i*-th element in the tree is changed to a value with hash *h*, the new root of the tree is *s*”? Describe (in words or pseudocode) what the prover would produce, and how would the verifier check that the proof is correct.**

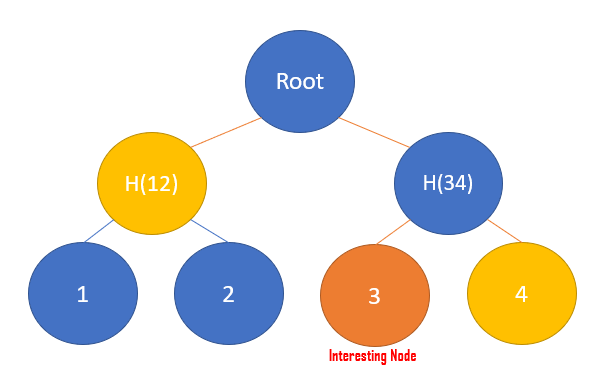
If the i-th element of the tree is changed to hash h then the prover should produce a new merkel root of the tree s. In order to prove this, the prover should provide the verifier with a new merkel proof of the new tree.

For creating a merkel proof, the prover should follow the following steps:

1. First create a new merkel tree with the new hash h.
2. Then compile the necessary hashes of the tree for verification process.
3. Merkel Proof will be combination of these necessary hashes, i-th hash (which was changed) and the root hash.

The process of compiling necessary hashes:

1. Start from the top of the tree.
2. If the subtree of the current node has no interesting node or if the current node is interesting, then take this hash value and do not continue on its subtree.
3. Else if the subtree has an interesting node, then continue with the subtree.



Hence in the above example the prover will give the following as the

**Merkel Proof-{H(12), 4, 3, Root}**

Now for checking that the proof provided by the prover is correct the verifier follows the following steps:

1. Start with the changed node.
2. With the help of the Hashes provided in the merkel proof create a pruned Merkel Tree Root.
3. And then verify that the root obtained by him is same as the root hash provided in the proof.

**Q2) How big is the proof, in the worst case?**

The worst case occurs when all the we have to provide a merkel proof for all the data points. In such a case we have to provide N hashes as the merkel proof.

Hence the worst space complexity of the merkel proof is O(N), where N is the number of transactions/data points from which the merkel tree is constructed.

**Q3) Why is it impossible to fool the verifier into accepting an invalid proof? What would a proof that fools the verifier imply about the chosen cryptographic hash function?**

It is nearly impossible for a prover to fool a verifier by providing him with an invalid proof as any slightest of the change in the merkel tree will change the merkel root and hence the proof becomes invalid. It is only possible to make fool of the verifier if the prover comes up with a change in the tree that does not changes the merkel root. But as the merkel trees are composed of cryptographic hash functions that are image-resistant so it is nearly impossible for the prover to fool the verifier.

If a proof is able to fool a verifier then it simply means that the cryptographic hash function used in the making of the merkel tree is not image-resistant and is not a good cryptographic hash function.