Lab1: Merkle Proof Verification

Due: 23:59 on Wed., Feb. 22, 2023 for students already in the class
Due: 23:59 on Tue., Feb. 21, 2023 for students who wish to sign-up or audit

---

Credit: This assignment is directly adapted from Stanford CS251, all credits goes towards the course. I simply loved the assignment!

In this assignment, you are tasked with writing a Python function that will generate a Merkle proof. The starter code, which you will find in a zip file named **sample.zip**, contains three files: **prover.py**, **verifier.py**, and **proof-for-leaf-95.txt**.

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/412c5536-51b5-4aaf-b85a-b4aa98746140/sample.zip

- **prover.py**: There is a list of a thousand strings that make up the roots of a Merkle tree. The `gen_merkle_proof()` function, which you will need to implement, will generate a Merkle proof for one of the leaves (leaf #743). The resulting proof will be written to a text file named **proof.txt**.
- **verifier.py**: You will find a hardcoded Merkle root for the Merkle tree generated by the prover. The verifier script loads the proof from **proof.txt** and verifies it with respect to the hardcoded Merkle root. You should not modify this file, but you should familiarize yourself with the `compute_merkle_root_from_proof()` function, which is the core of the verifier.
- **proof-for-leaf-95.txt**: An example Merkle proof that is accepted by the verifier. Your goal is to generate a proof file like this for leaf #743.

After you have successfully implemented the `gen_merkle_proof` function in **prover.py**, running both **prover.py** and **verifier.py** consecutively should produce the following output:

```
$ python prover.py
I generated 1000 leaves for a Merkle tree of height 10.
I generated a Merkle proof for leaf #743 in file proof.txt

$ python verifier.py
Have hardcoded root of committed Merkle tree.
I verified the Merkle proof: leaf #743 in the committed tree is "data item 743".
```

It is recommended that you test the proof's validity by changing one character in the **proof.txt** file and verifying that the verifier rejects the proof.

There are lots of implementation of Merkle trees on the internet, however, you'll gain much deeper understanding of the concepts if you code it yourself.

Plus it's more fun! ;)