A *newish* approach to Quantum Hash Function

Overview:

The project implements a quantum hash function that ingests input data of size 2^N bytes and produces a quantum hash of the same size, using a parameterized quantum circuit. It explores a novel approach to hashing where quantum state evolution and expectation values determine the final output.

This project extends on the existing quantum hash function provided in the github repo. This new function introduces ability to generate 2^N bytes of hash for 2^N bytes of input. To accomplish this, various iterations of quantum circuits have been utilized. For $n + 1th$ time of running the quantum circuit, output from nth time has been used as input. This introduces complexity in the hash function.

Key Features

- Input Size: Accepts input data as bytes of size (2^N).
- Output Size: Produces a hash output of the same size as the input.
- Quantum Circuit: Utilizes parameterized quantum gates (`rx`, `ry`, `rz`) and controlled NOT (`cx`) gates.
- Fixed-Point Conversion: Converts floating-point expectation values to fixed-point integers for output generation.
- Circuit Information: Optionally returns details about the quantum circuit, such as depth, qubit count, and gate count.

Function Descriptions

`toFixed(x: float, fraction_bits: int) -> int`

Converts a floating-point number to a fixed-point integer representation.

- Parameters:
  - `x`: The floating-point number to convert.
  - `fraction_bits`: The number of fractional bits for fixed-point representation.

- Returns: Fixed-point integer representation of `x`.

`qhash(x: bytes, returnCircuitInfo: bool = False) -> bytes | tuple`

Generates a quantum hash for the given input data.

- Parameters:
  - `x`: Input data as bytes of size (2^N).
  - `returnCircuitInfo`: If `True`, returns additional information about the quantum circuit.
- Returns:
  - If `returnCircuitInfo` is `False`: The quantum hash as bytes.
  - If `returnCircuitInfo` is `True`: A tuple containing the quantum hash and circuit information.
- Circuit Details:
  - Number of Qubits is basically N.
  - Number of Layers: 4 layers of parameterized gates.
  - Parameterized Gates: `rx`, `ry`, `rz` gates with angles derived from the input data.
  - Entanglement: Achieved using `cx` gates between adjacent qubits.
- Hash Generation:
  - Input data is used to parameterize the quantum circuit.
  - The circuit's `statevector` is used to compute expectation values of the Pauli-Z operator.
  - Expectation values are converted to fixed-point integers and then to bytes.
- Feedback Loop
  - The newly generated output bytes are used as the next round's `input_data` to increase randomness and depth. This process continues iteratively until enough bytes are collected.
- Optional Circuit Information:
  - `qubits`: Number of qubits in the transpiled circuit.
  - `depth`: Depth of the transpiled circuit.
  - `gate_count`: Count of each gate type in the transpiled circuit.

`check_io_size(input_data: bytes, hash_data: bytes) -> bool`

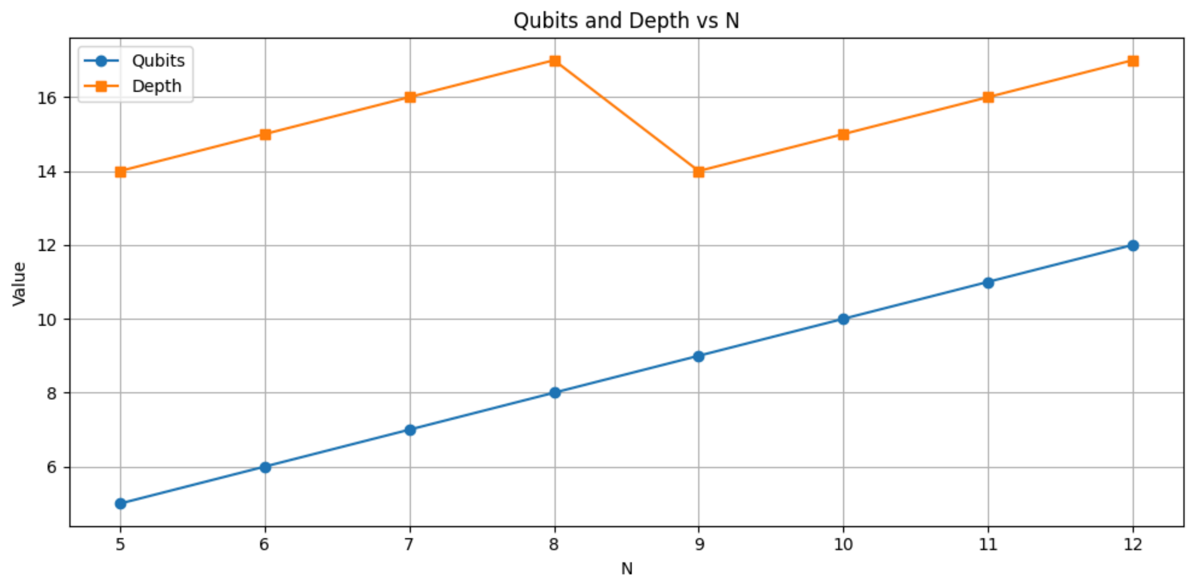
Checks if the input data size matches the hash output size.

- Parameters:
  - `input_data` : The original input data as bytes.
  - `hash_data` : The generated hash data as bytes.
- Returns: `True` if the sizes match, `False` otherwise.

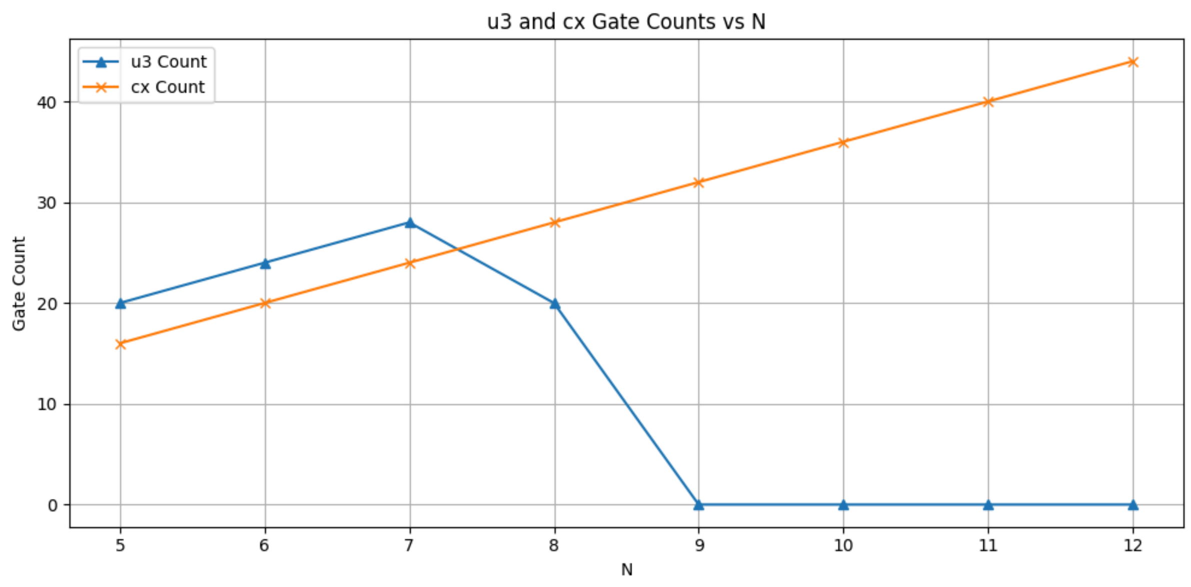
Analyzing the Quantum Hash Function:

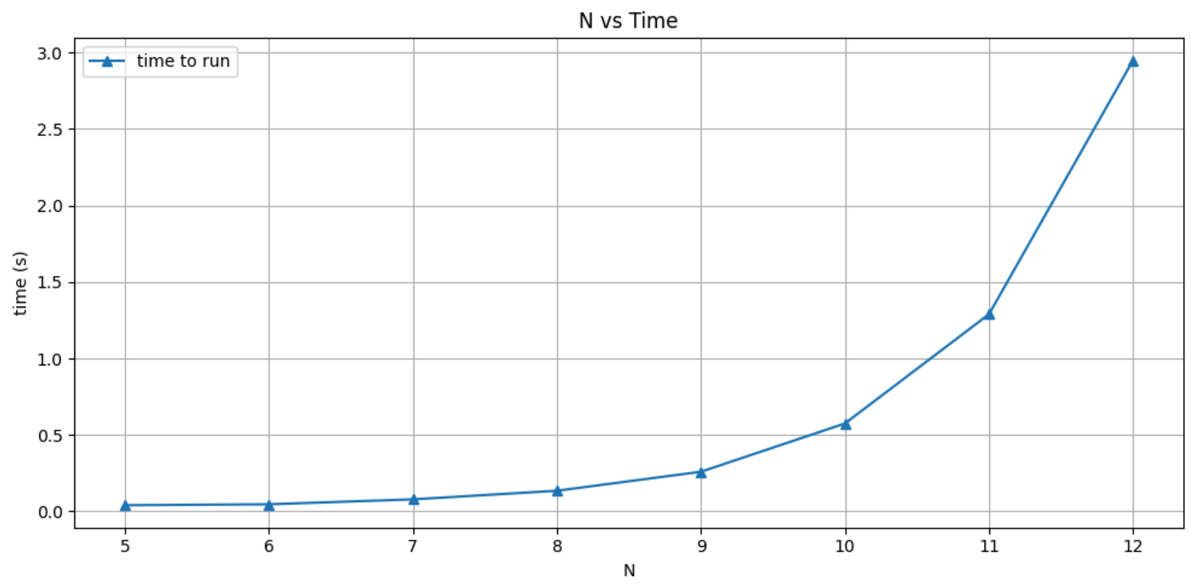
For N = 7, below are some of the metrics which can provide good overview of the quantitative side of the quantum hash function,

1. **Determinism**: The Quantum Hash Function is deterministic, and is verified over good range of data. Also, this property is explainable by the structure of our quantum circuit(there are no random steps used).
2. **Entropy Preservation:** Average entropy per byte is, $5.65/8$
3. **Preimage Resistance:** it would be so difficult to find preimage to this function, if not impossible. For input size of 2^7, and 1000 trials, any preimage to the hash function was no found.
4. **Collision Resistance**: for 500 samples, and same input size, there were no collisions.
5. **Computational Complexity:** for this consider the below graphs, they are presented for data from range of $N = 5$, to $N = 12$



The first figure shows the relation between $NumberofQubits$ and $Depth$ of quantum circuit. It can be asserted from the graph that $depth$ may be periodic with N. and $NoofQubits$ is obviously in linear relation with N.





This is the most important graph. Since it shows that $time$ to execute the hashing algorithm grows exponentially with N. This brings in computational difficulty.

💡 All of the above claims can be verified by various metrics available in `analysis.py`

Project by - Samradh

Loosely Inspired by,

1. https://arxiv.org/pdf/2408.03672
2. https://arxiv.org/pdf/2310.17439
3. https://www.nature.com/articles/s41598-023-33119-w