**Properties:**

1. Inverse QFT, will always return the original qubit states (wrote a test for this, works as intended)
2. The first qubit can be in the states after QFT algorithm has been applied, and the second can be in the states. The problem with this, is that it may be difficult to assert the phase of the qubits.

The phase of the output is dictated also by the length in qubits. QFT on qubits ‘1’ is different to QFT on ‘0001’.

The equation for the phase is where ‘’ is the real value in qubits, and n is the position of the qubit (starting from 1)

So, for ‘0001’ we expect the phases to be:

for the first qubit (on the left)

90 for the second

45 for the third

22.5 for the fourth

1. A simpler property that can be used to do with phase, no matter the qubit value, we know that the phase for each qubit must be a multiple of certain amount of **c,**

**c**

4.The LSB qubit will always be less than

**Properties (new):**

1.

Precondition:

Vector of qubits x0 …. xn-1

Operation:

InvQFT(QFT(x0, ..., xn-1))

Output:

assertEqual((x0, ..., xn-1), (x'0, ...., x'n-1)

2.

Precondition:

Vector of qubits x0 … xN-1 such that \forall i : 0 <= i < n . x\_i = |0> or x\_i = |1>

Where binary total of x0 … xN-1 = T

Operation:

QFT(x0, ..., xN-1)

Output:

assertPhase((x0,…, xN-1), (**c** ,…, **c**)

3.

Precondition:

Vector of qubits x0 …. xN-1

Operation:

QFT(x0, ..., xN-1)

Output:

assertTrue(estimatePhase(x0) MOD **c** )

.

.

.

assertTrue(estimatePhase(xN-1) MOD **c** )

4.

Precondition:

Vectors of qubits x0 …. xN-1, y0 …. yN-1

Where ( x0 … xN-1 > y0 … yN-1 )

Operation:

QFT(x0, ..., xN-1)

QFT(y0, ..., yN-1)

Output:

assertTrue(estimatePhase(xN-1) > estimatePhase(yN-1))

6.

Precondition:

x0 …. xN-1 – ,

y0 …. yM-1 –

Where

Operation:

QFT(x0, ..., xN-1)

QFT(y0, ..., yN-1)

Output:

assertTrue(estimatePhase(xN-1) > estimatePhase(yN-1))

(if max value and different lengths, higher length last qubit should be bigger value)

7.

(if T MOD N-1 for a qubit == 0, set the phase should be 0)