Microsoft IONQ iQuHACK 2022

PurpleQAT

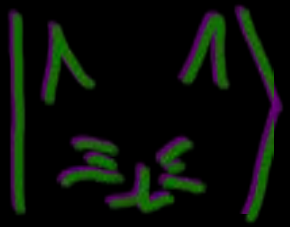
Teaching Quantum Circuits with Boxes

Our Motivation

Use quantum logic itself as a gaming mechanism for education

In order to explain the applications of an algorithm it is necessary to explain what a quantum circuit is and what better with video games. We can motivate the people start in the quantum area with videogames and they find relations with this videogame.

We replace the theory for image of cats and boxes.



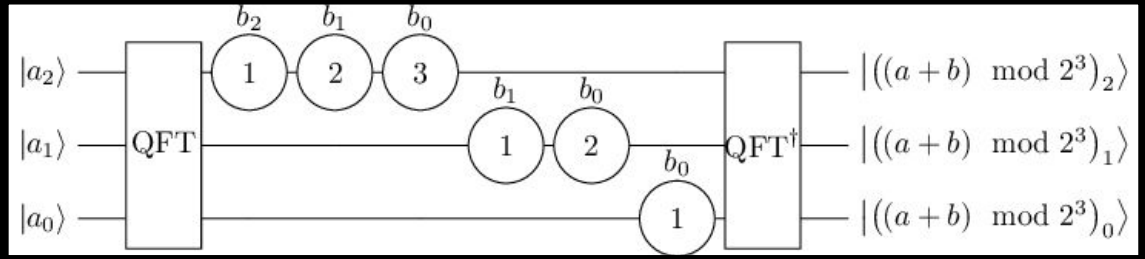


Image obtained from:
https://www.researchgate.net/figure/Modified-Drapers-adder-example-for-3-qubit-register-a-and-3-bit-classical-constant-b_fig3_340271339

First problem

Quantum Random Number Generator (QRNG)

Using Q# and the idea of having a random number generator using a quantum circuit to find numbers up to 32, this was based on the idea that there is noise in the computer such as the generation of a noisy model.

```
open Microsoft.Quantum.Arrays;
open Microsoft.Quantum.Measurement;
operation RandomNumber() : Result[] {
    use register = Qubit[5] {
        // Set qubits in superposition.
        for index in 0 .. 10  {
        ApplyToEachA(H, register);
        CNOT(register[0], register[1]);
        CNOT(register[0], register[2]);
        CNOT(register[0], register[3]);
        CNOT(register[0], register[4]);
        CNOT(register[1], register[2]);
        CNOT(register[1], register[3]);
        CNOT(register[1], register[4]);
        CNOT(register[2], register[3]);
        CNOT(register[3], register[4]);
        CNOT(register[3], register[4]);
        X(register[0]);
        }
        return ForEach(MResetZ, register);
    }
}
```

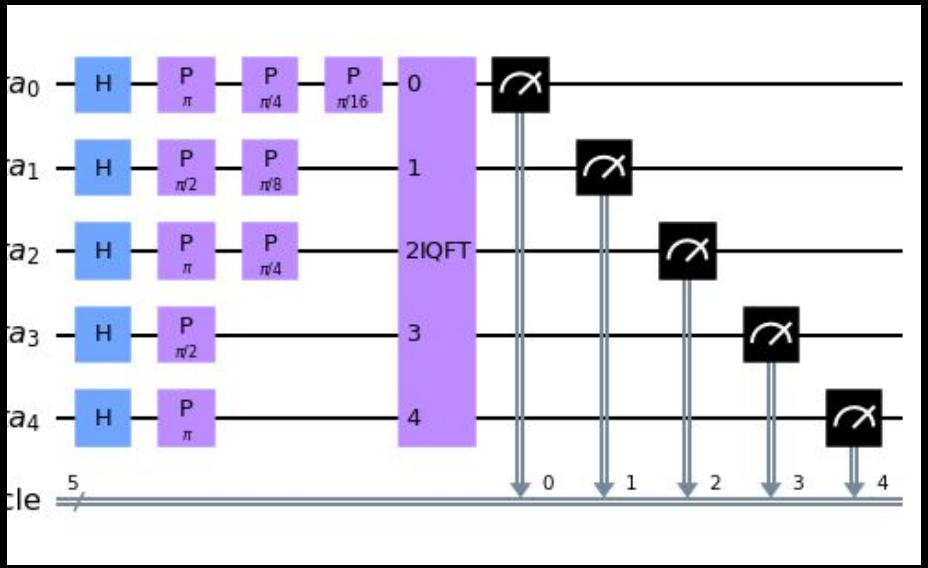
Second problem

Draper Adder and substract

Understand the concepts of

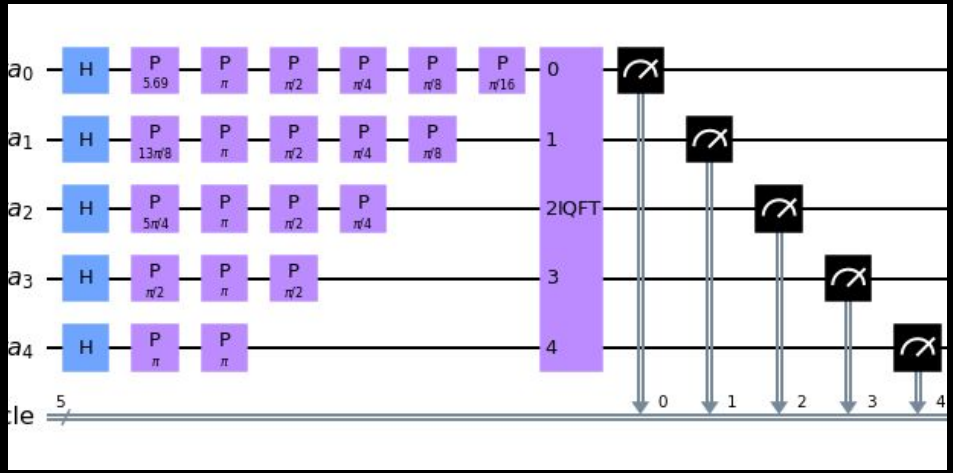
- Amplitude Amplification
- Quantum Fourier Transform
- Arithmetic Operations

And using the ionq.qpu



Quantum gaming mechanism

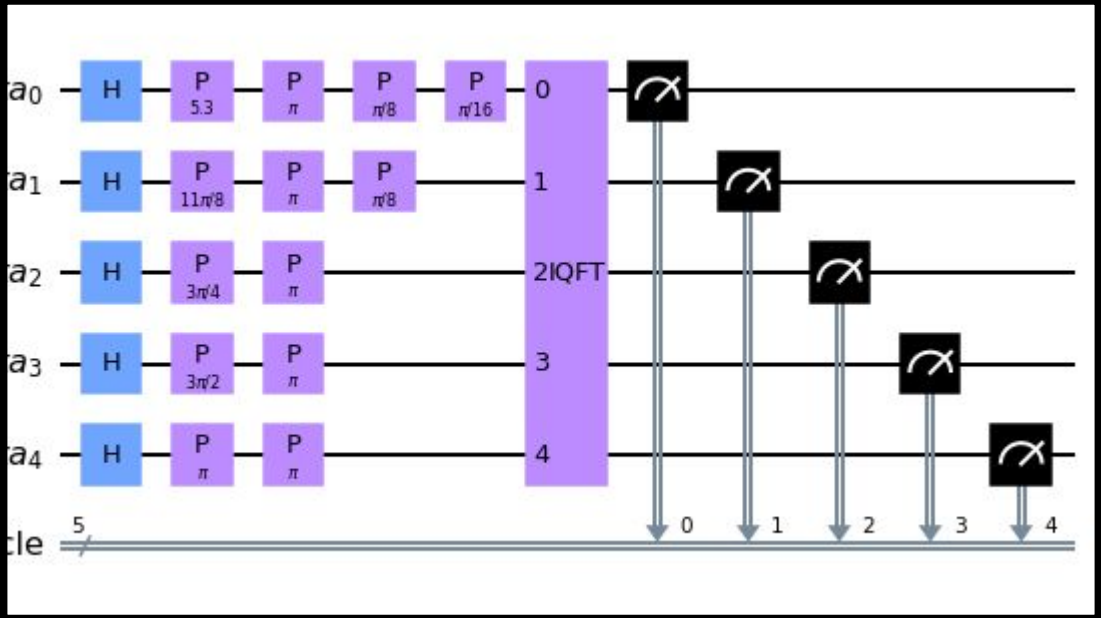
This is the all quantum circuit, we can consider or not certain P gates.



Level 1

Consider all the cats are green or the qubits are in |0> and we need find the correct combination of green and purple cats. That means a combination of |0> and |1> states.

Example the output is |11001>

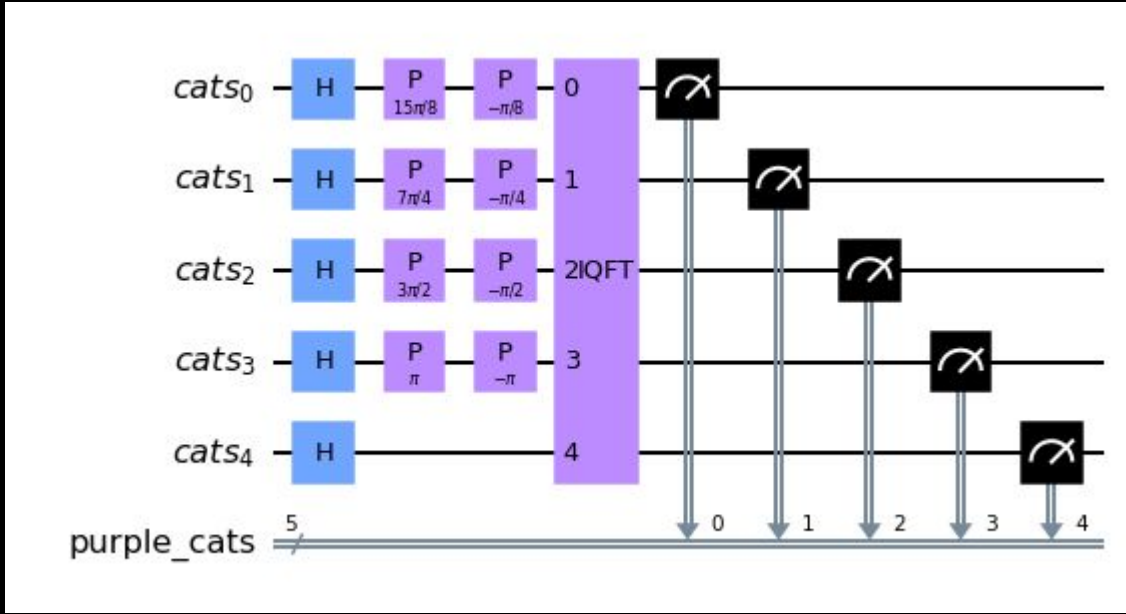


Level 2

Initial state = |11110>

We need add |00010>

Example the output is |11100>



Mechanics(How does it work)

We start out with the cats in superpostion (green cats) which are the qubits.

We use the Draper Method for addition and subtraction.

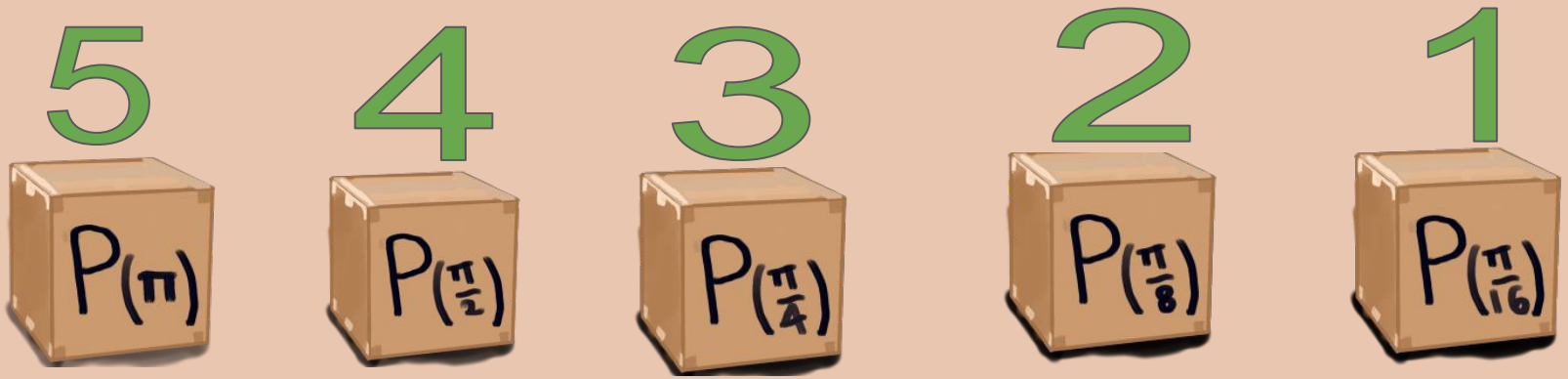
By using gates in the circuits we can change the color of the cats (i.e. the states of the cats).

The goal is to find the original cats(purple) from the changed cats(green).

RULES

In order to find the right cats we have a certain number of boxes with a unique rotation.

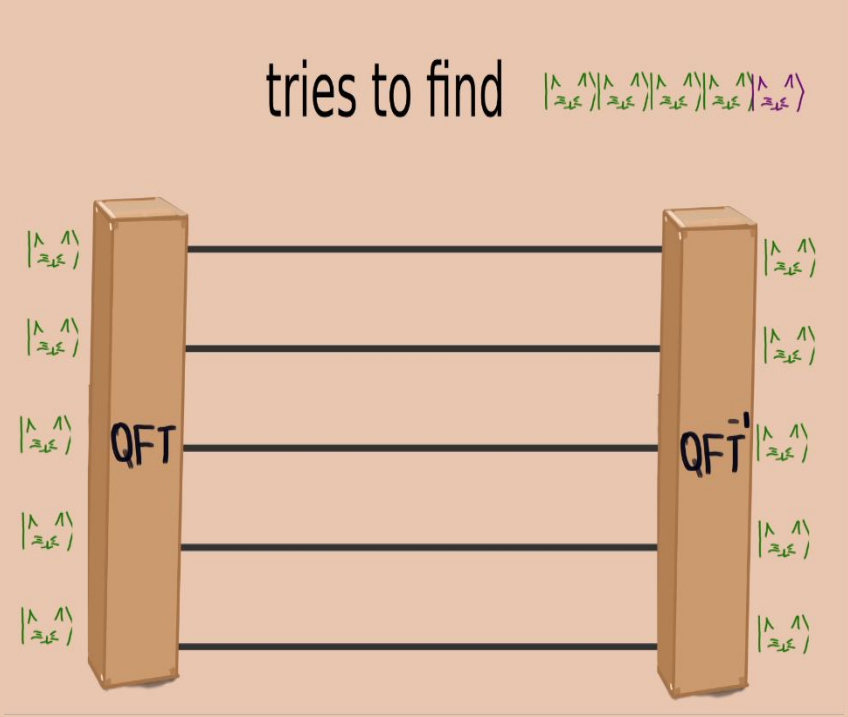
You can choose to use more or less boxes depending on the structure of the circuit.

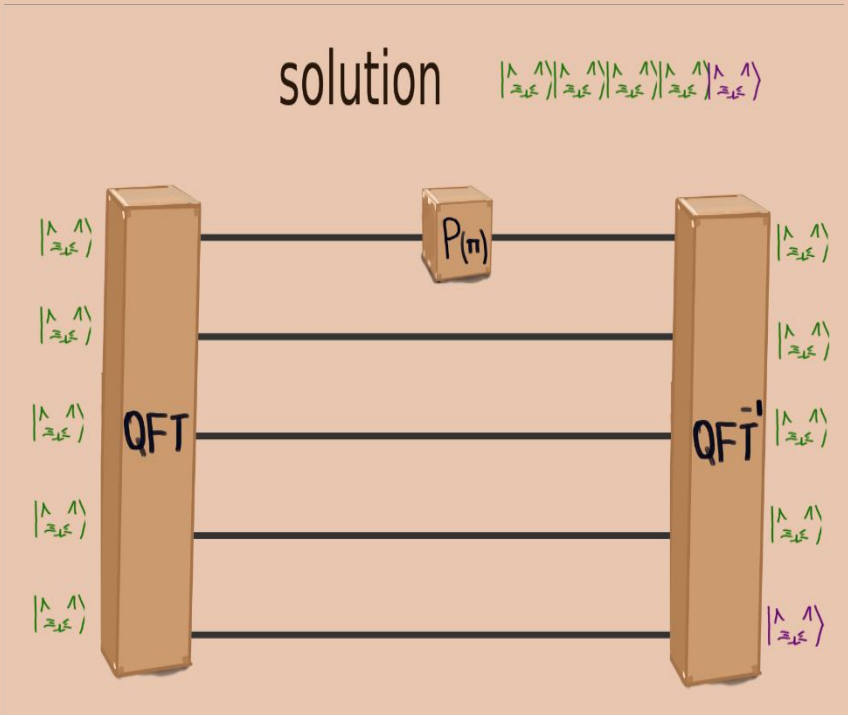


Hints

There are configurations known to find the purple cats.

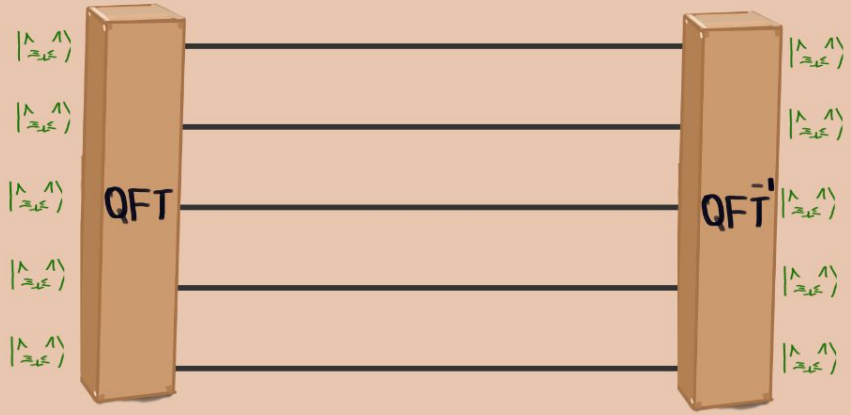
HINT(1)



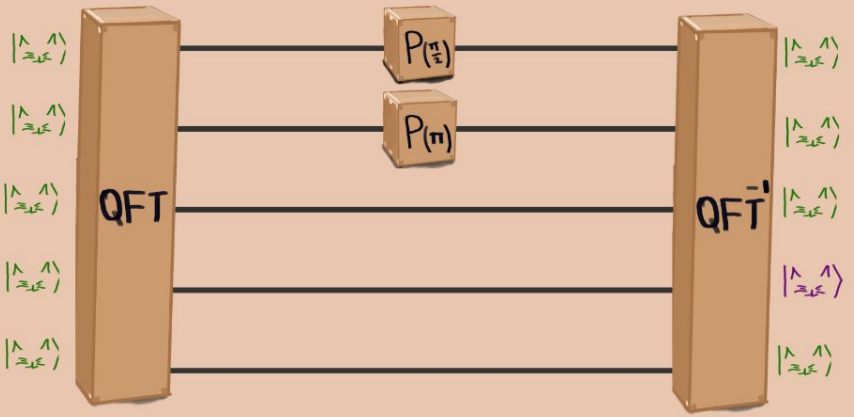


HINT(2)

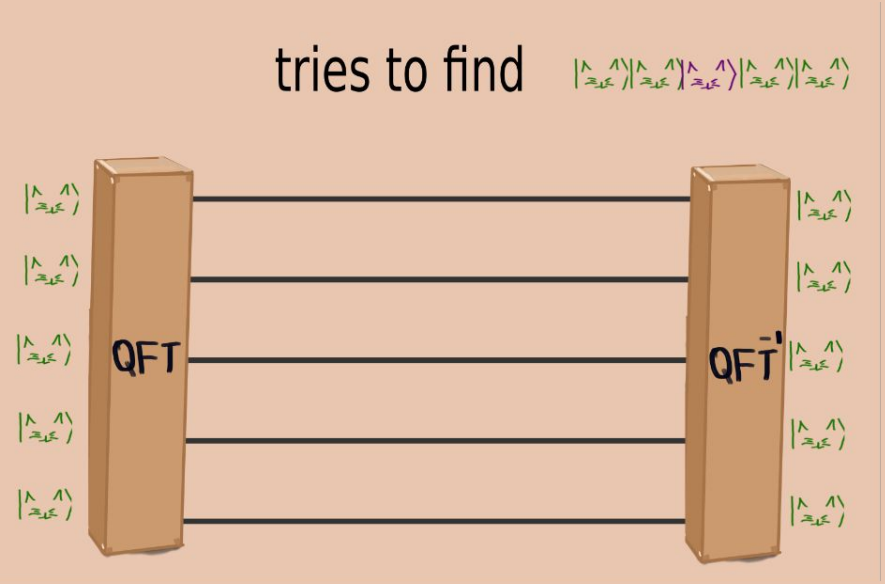
tries to find

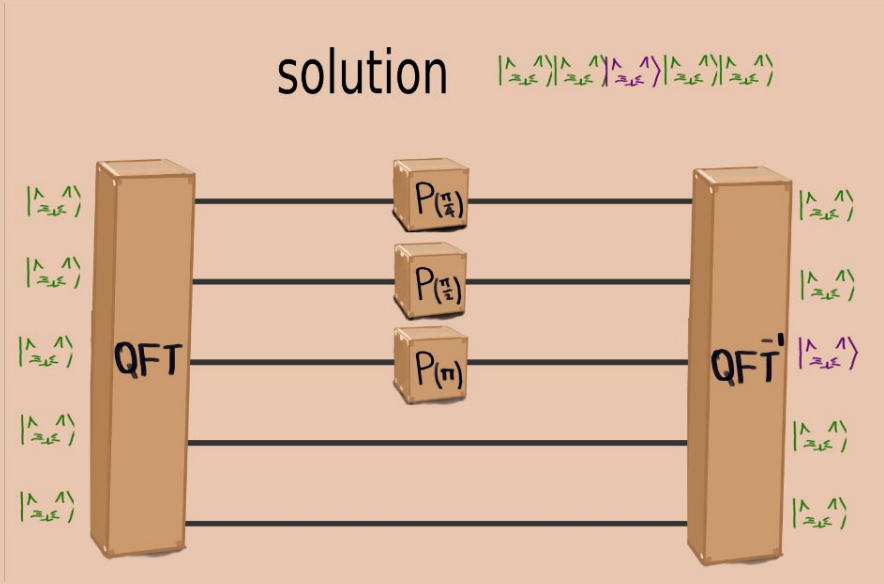


solution



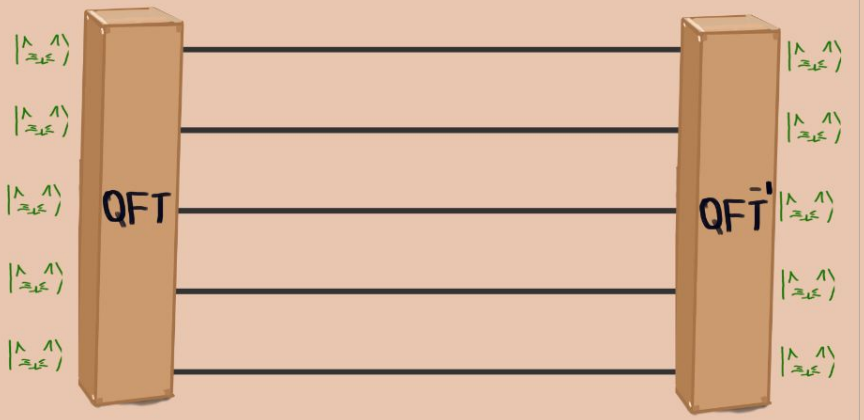
HINT(3)



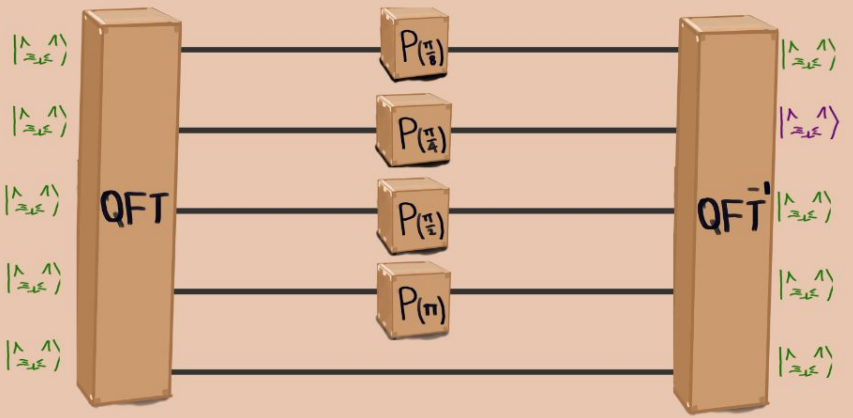


HINT(4)

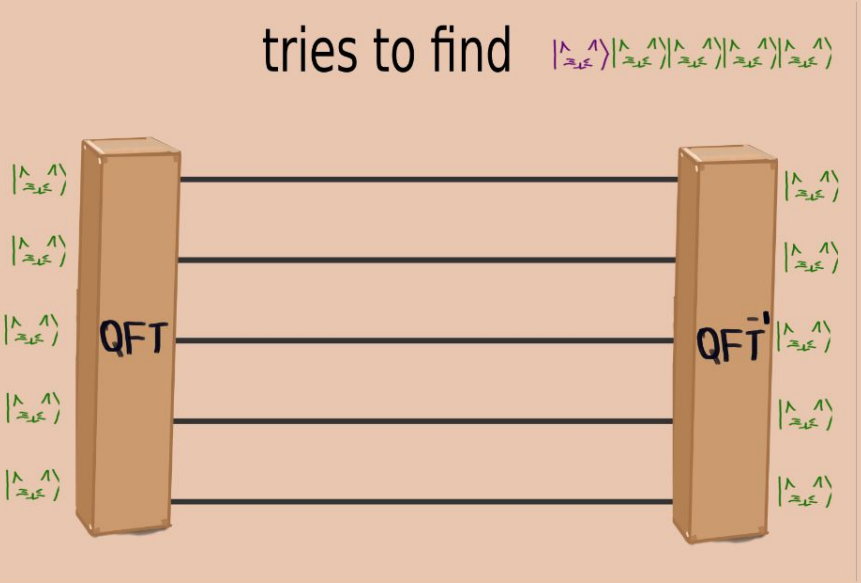
tries to find

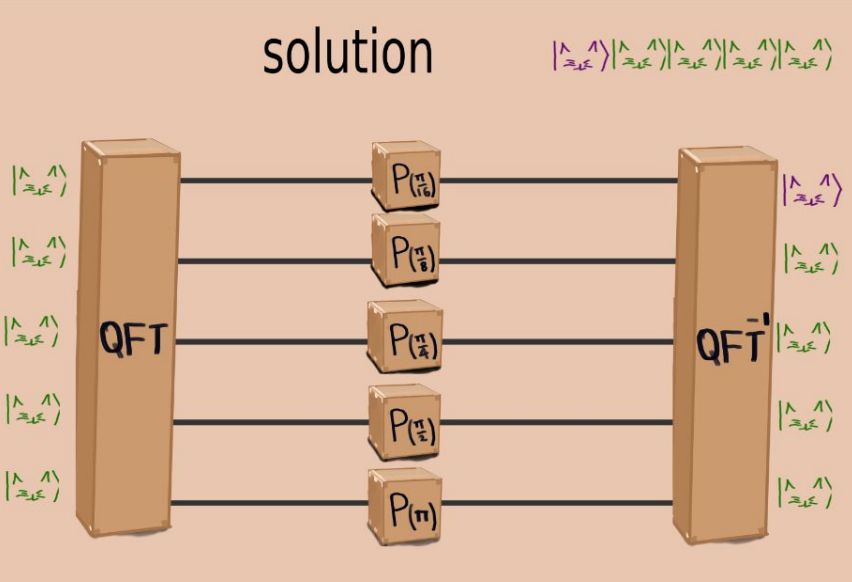


solution



HINT(5)





Screenshot(First Modality)

```
qc = find_purple_cats(pi,pi_2,pi_4,pi_8,pi_16)
counts = execute(qc,provider.get_backend('ionq.simulator')).result().get_counts()

sol=list(counts.keys())[0]

print("your answer is ",sol)
if purple_cats_numer == sol:
    print("You Got it")
else:
    print("Try again")
```

✓ 53 sec

```
.....your answer is  00100
You Got it
```

simulation

Real quantum computer

using a ionq qpu

we compare the result with a real hardware

```
qc = find_purple_cats(pi,pi_2,pi_4,pi_8,pi_16)

target_basis = ['rx', 'ry', 'rz', 'h', 'cx']
decomposed = transpile(qc,
                       basis_gates=target_basis,
                       optimization_level=0)

qpu_backend = provider.get_backend("ionq.qpu")
qpu_job = qpu_backend.run(decomposed, shots=1024)
job_id = qpu_job.id()

# Get the job results (this method also waits for the Job to complete):
import operator
result = qpu_job.result()
counts = {format(n, "03b"): 0 for n in range(8)}
counts.update(result.get_counts(qc))
sol= max(counts.items(), key=operator.itemgetter(1))[0]
print("your answer is ",sol)
if purple_cats_numer == sol:
    print("You Got it")
else:
    print("Try again")
```

✓ 24 min 2 sec

Screenshot(Second Modality)

```
# init variables
pi = [0]*5
pi_2 = [0]*4
pi_4 = [0]*3
pi_8 = [0]*2
pi_16 = [0]


# select the values


pi[3] = -1
pi_2[2] = -1
pi_4[1] = -1
pi_8[0] = -1
```

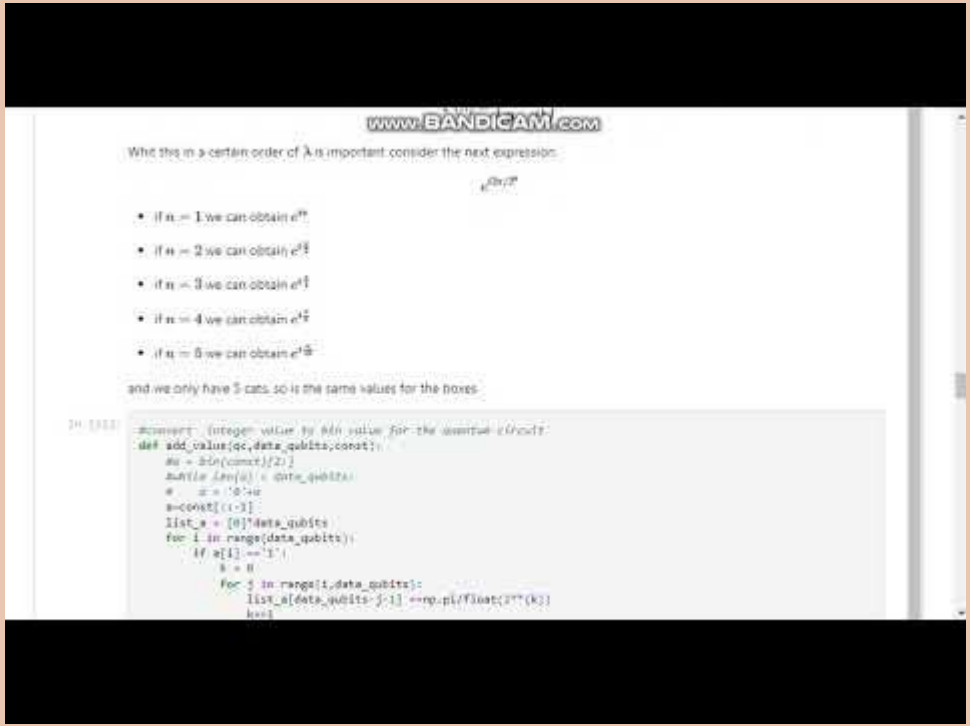
```
qc = find_purple_cats(pi,pi_2,pi_4,pi_8,pi_16,init_state)
counts = execute(qc,provider.get_backend('ionq_qpu')).result().get_counts()
sol=list(counts.keys())[0]

print("your answer is ",sol)
if purple_cats_numer == sol:
    print("You Got it")
else:
    print("Try again")
```

✓

```
.....your answer is  10110
You Got it
```

Demo

https://youtu.be/zL1ZSIpYx2M



Summary

We **developed a quantum random number generator** for the target value of the game and the game mechanism is based on a quantum computation algorithm that takes 3 different concepts such as A**mplitude Amplification, Quantum Fourier Transform and Arithmetic Operations,** in order to relate different concepts in something more superficial and that anyone can practice.

Its purpose is to **introduce different concepts and characteristics of quantum computation** in an indirect way.

To be able to **abstract complex concepts so that only relationships** are sought with the little information and motivate **the player to interact with the different combinations** that are established.

Next Steps

Improving

We need to make this video game in an interactive way where animations and transitions show the way to play in order to make it more attractive and friendly for the players.

Extending

More qubits can be used and with multiple states or possibilities to show other features such as superposition or Grover's algorithm.

Developing

Due to the time constraints of 26 hours, it can be improved in the future to be applicable to an environment suitable for the development and design of video games.

References

Preskill, John. (2018). Quantum Computing in the NISQ era and beyond. Quantum. 2. 10.22331/q-2018-08-06-79.

Draper, Thomas. (2000). Addition on a Quantum Computer.

Ruiz Pérez, Lidia & Garcia-Escartin, Juan Carlos. (2017). Quantum arithmetic with the Quantum Fourier Transform. Quantum Information Processing. 16. 10.1007/s11128-017-1603-1.

https://docs.microsoft.com/en-us/azure/quantum/user-guide/libraries/standard/algorithms

Suau, Adrien & Staffelbach, Gabriel & Calandra, H.. (2020). Practical Quantum Computing: solving the wave equation using a quantum approach.

https://docs.microsoft.com/en-us/azure/quantum/how-to-python-qdk-local

https://docs.microsoft.com/en-us/azure/quantum/user-guide/machines/noise-simulator#:~:text=You%20can%20modify%20the%20noise%20model%20used%20in,then%20access%20the%20noise%20model%20by%20using%20get_noise_model%3A

https://iong.com/docs/get-started-with-qiskit

https://en.wikipedia.org/wiki/Pseudorandom_number_generator

https://docs.microsoft.com/en-us/azure/quantum/user-guide/libraries/standard/algorithms

https://docs.microsoft.com/en-us/azure/quantum/how-to-python-qdk-local