The False Proof

500 points

Backstory

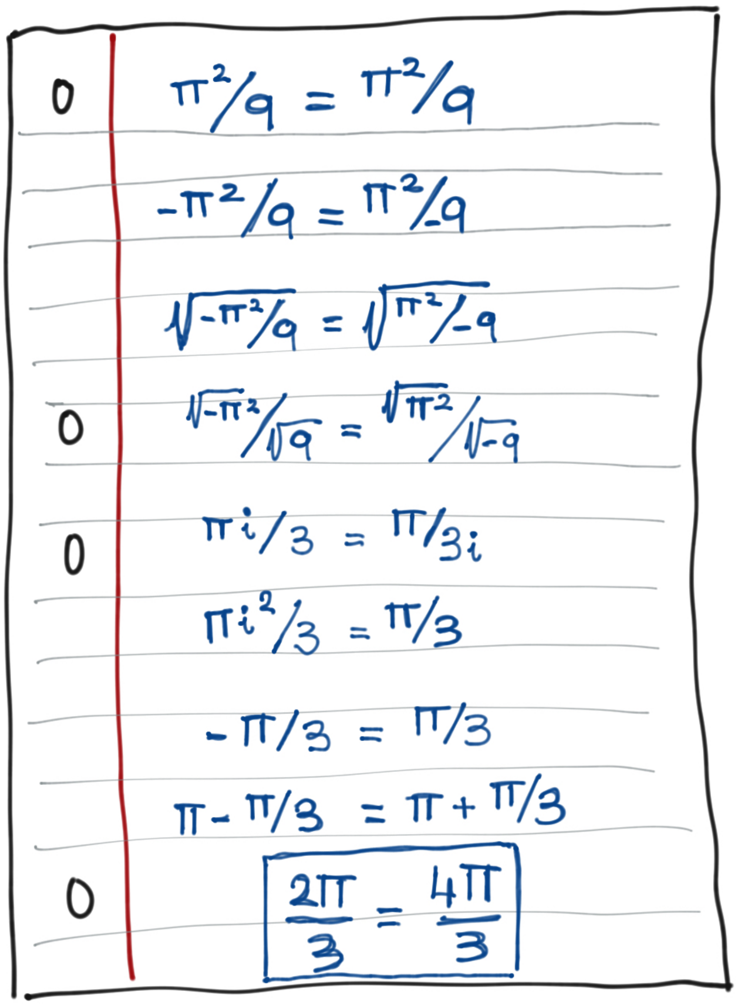
Trine's Designs is very community-oriented, and customers like to write letters to the staff to get help with a variety of issues. A mysterious person has sent a non-sensical mathematical proof about complex numbers. No one at the company is good at complex analysis, but Zenda and Reece set out to use a quantum circuit to prove the anonymous customer wrong.

One-shot proof

In the 17th century, a nice custom spread among intellectuals: challenging each other to solve numerical problems and puzzles. These challenges were sent through letters and, thanks to this, mathematics expanded and gained a lot of popularity.

It seems that this tradition is making a comeback because the other day Reece received the following letter at the office:

"Hello Trine team, I have just come to a discovery of great importance, I have just proved that $\frac{2\pi}{3} = \frac{4\pi}{3}$. I attach a simple proof. If you think I'm wrong, prove it!*"



$$\pi^2/9 = \pi^2/9$$
$$-\pi^2/9 = \pi^2/-9$$
$$\sqrt{-\pi^2/9} = \sqrt{\pi^2/-9}$$
$$\sqrt{-\pi^2}/\sqrt{9} = \sqrt{\pi^2}/\sqrt{-9}$$
$$\pi i/3 = \pi/3i$$
$$\pi i^2/3 = \pi/3$$
$$-\pi/3 = \pi/3$$
$$\pi - \pi/3 = \pi + \pi/3$$
$$\boxed{\frac{2\pi}{3} = \frac{4\pi}{3}}$$

Actually, the proof is very convincing, but Reece turns to quantum computing to show that this is not so (for fun, if you like math: find the line where the mistake is!). To do this, Reece manages to show that $R_Y(\frac{2\pi}{3})$ is different from $R_Y(\frac{4\pi}{3})$. In fact, they are so different that a single shot is enough to distinguish them!

This will be your objetive for this challenge! We are going to give you a gate U, which you know is either $R_Y(\frac{2\pi}{3})$ or $R_Y(\frac{4\pi}{3})$. Your task will be to build a quantum circuit containing U to unambiguously determine which of the two gates it is using only *one shot*.

To do this, you will have access to a two-qubit circuit. You can use U as many times as you want, but one thing is important: the solution must guarantee with **100% probability** which of the two options it is.

Note: even if you do not know U, you are allowed to use `qml.ctrl` over U if necessary.

▼ **Epilogue**

After successfully debunking the proof, Zenda and Reece are enjoying a well-deserved rest from office hijinks when Doc Trine, the boss herself, appears from the basement laboratory. "Well, it seems you have passed my little math test! Please, step into my office." Her office is the basement, and they filled with all sorts of odds and ends: a machine whose only function is to turn itself off, an anti-toaster for making old bread fresh, some sort of tiny imp lugging around two chambers and looking for work. Trine looks at them: "I think you might be excited by my latest invention. It promises to save us all a lot of time!"

*Read on in **Bending Bennett's Laws**.*

Challenge code

On one hand you are asked to complete `circuit` (you only need to apply gates). On the other hand you must complete `process_output`, which will take the output of `circuit` (a vector of dimension two where each term can take the value 0 or 1) and will return 2 if $U = RY(2\pi/3)$ or 4 if $U = RY(4\pi/3)$.

Output

The circuit function will receive the gate U that you are asked to determine. To judge whether your circuit works as expected, we will randomly send 5000 different examples and check that they always classified correctly. Therefore, in this question there are no public or private test cases. They are randomly generated. Good luck!

Code

Help

```
import json
import pennylane as qml
import pennylane.numpy as np
```

```python
dev = qml.device("default.qubit", wires=2, shots=1)
dev.operations.add("op")
dev.operations.add("C(op)")

```

```python
@qml.qnode(dev)
def circuit(U):
    """This will be the circuit you will use to determine which
    Remember that only a single shot will be executed.

    Args:
        U (qml.ops): It is the gate to discriminate between  RY(

    Returns:
        (numpy.tensor): Vector of two elements representing the
    """
    # Put your code here #
    # to use U,  call 'U(wires = <wire where you want to apply t
    # to use Control-U, call 'qml.ctrl(U, control = <control wir
    return qml.sample(wires=range(2))

def process_output(measurement):
    """This function processes the output of the circuit to disc

    Args:
        measurement (numpy.array): Output of the previous circui

    Returns:
        (int): return 2 or 4 depending on the associated RY gate
    """
    # Put your code here #

# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:
    return None

def check(solution_output: str, expected_output: str) -> None:
    numbers = 2 * np.random.randint(1, 3, 5000)

    def U2(wires):
        class op(qml.operation.Operator):
            num_wires = 1

            def compute_decomposition(self, wires):
                raise ValueError("You cannot decompose this gate

            def matrix(self):
                return qml.matrix(qml.RY(2 * np.pi / 3, wires=3)

        op(wires=wires)
        return None

    def U4(wires):
        class op(qml.operation.Operator):
            num_wires = 1

            def compute_decomposition(self, wires):
                raise ValueError("You cannot decompose this gate

            def matrix(self):
                return qml.matrix(qml.RY(4 * np.pi / 3, wires=3)

        op(wires=wires)
        return None

    output = []
    for i in numbers:
        if i == 2:
            U = U2
        else:
            U = U4
        out = circuit(U)
        output.append(process_output(out))

    assert np.allclose(
        output, numbers, rtol=1e-4
    ), "Your circuit does not give the correct output."

```

```python
test_cases = [['No input', 'No output']]
```

```python
for i, (input_, expected_output) in enumerate(test_cases):
    print(f"Running test case {i} with input '{input_}'...")

    try:
        output = run(input_)

    except Exception as exc:
        print(f"Runtime Error. {exc}")

    else:
        if message := check(output, expected_output):
            print(f"Wrong Answer. Have: '{output}'. Want: '{expe

        else:
            print("Correct!")
```

Copy all

Submit

Open Notebook

Reset