QHack

Quantum Coding Challenges

🚀 CHALLENGE COMPLETED

View successful submissions

Jump to code — Collapse text

The Change of Qubit

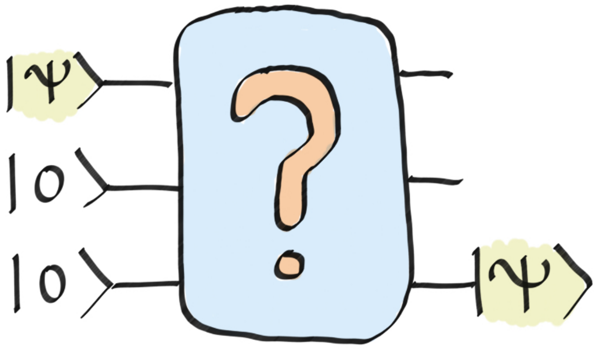
300 points

Backstory

Zenda needs to send an email to Reece through Trine's Designs' Quantum Area Network. The network is a quantum circuit that simply swaps qubits between wires. However, a virus seems to be interfering with proper communication within the network. Zenda can't get rid of the virus, but she has figured out what it's doing. Let's help her reprogram the network to get around the issue.

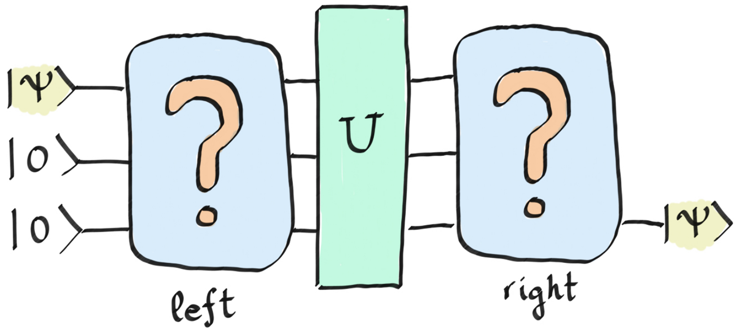
More than just a SWAP

This challenge's statement is very simple, but solving it may not be so easy. The goal will be to move a state $|\psi\rangle$ from one qubit to another, as shown in the figure:



Easy, isn't it? Well, we are going to complicate it a little bit. We will not allow any direct connections between the first and last qubits, so you will have to use the middle qubit to pass the information from one qubit to another.

The exercise has become more challenging, but we still want to complicate it further! In addition to all of the above, we are going to generate a virus operator U and place it in the middle of the circuit, like so:



In this case, U consists of one `PauliX` gate that we place randomly in one of the three qubits (each time, the gate is generated in a different wire). You should be able to complete the circuit so that, regardless of the randomly generated operator U, you are able to move the state from the first qubit to the last qubit.

Challenge code

In this challenge, you will only be asked to complete the `circuit_left` and `circuit_right` operators to meet the objective of the statement.

Input

To encode the initial state $|\psi\rangle$, we will use a $U3$ gate. The input will be the three parameters (`alpha`, `beta`, `gamma`) associated with this gate.

Output

In this case, the output is the measurement result of the last qubit with respect to an observable. In this way, we check if it coincides with the expected state. Good luck!

Code

Help

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

```python
def circuit_left():
    """
    This function corresponds to the circuit on the left-hand si
    description. Simply place the necessary operations, you do r
    """

def circuit_right():
    """
    This function corresponds to the circuit on the right-hand s
    description. Simply place the necessary operations, you do r
    """

```

```python
def U():
    """This operator generates a PauliX gate on a random qubit""
    qml.PauliX(wires=np.random.randint(3))


dev = qml.device("default.qubit", wires=3)

@qml.qnode(dev)
def circuit(alpha, beta, gamma):
    """Total circuit joining each block.

    Args:
        alpha (float): The first parameter of a U3 gate.
        beta (float):The second parameter of a U3 gate.
        gamma (float): The third parameter of a U3 gate.

    Returns:
        (float): The expectation value of an observable.
    """
    qml.U3(alpha, beta, gamma, wires=0)
    circuit_left()
    U()
    circuit_right()

    # Here we are returning the expected value with respect to a
    # the choice of observable is not important in this exercise

    return qml.expval(0.5 * qml.PauliZ(2) - qml.PauliY(2))

```

```python
# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:
    angles = json.loads(test_case_input)
    output = circuit(*angles)
    return str(output)

def check(solution_output: str, expected_output: str) -> None:

    solution_output = json.loads(solution_output)
    expected_output = json.loads(expected_output)
    assert np.allclose(
        solution_output, expected_output, atol=2e-1
    ), "The expected output is not quite right."

    ops = circuit.tape.operations

    for op in ops:
        assert not (0 in op.wires and 2 in op.wires), "Invalid c

    assert circuit.tape.observables[0].wires == qml.wires.Wires(

```

```python
test_cases = [['[2.0,1.0,3.0]', '-0.97322'], ['[-0.5,1.2,-1.2
```

```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