







QHack

Quantum Coding Challenges

-  RANK
-  TEAM
-  CHALLENGES
-  SUBMISSIONS
-  SUPPORT

 CHALLENGE COMPLETED

[View successful submissions](#)

Jump to code

Collapse text

A Halfway-Decent Photocopier

400 points

Backstory

Trine is getting bored. "So, we've finished the usual *Laws of Infodynamics*. Let's make things more interesting!" She shows Zenda and Reece to the office photocopier. "I figured out a way to turn this into a quantum resource! Pretty cool huh?" Zenda and Reece look at each other, puzzled as ever by Trine's unconventional ideas about office equipment. Trine pats the photocopier. "Yup, this old thang can photocopy a basis state. You can use it to make Bell pairs! In fact, we can introduce the photocopier into our superdense and teleportation protocols in such a way that it turns infodynamic inequalities into equalities. We'll find that a photocopier is halfway between a qubit and an entangled bit! I always said this was a halfway-decent photocopier." Zenda and Reece shrug and start feeding qubits into the machine.

Coherent protocols

Zenda and Reece are having a bit too much fun feeding qubits into the photocopy machine. It's not very good at photocopying qubits, but it can copy basis states into new registers in the following way

$$|j\rangle \mapsto |j\rangle|j\rangle,$$

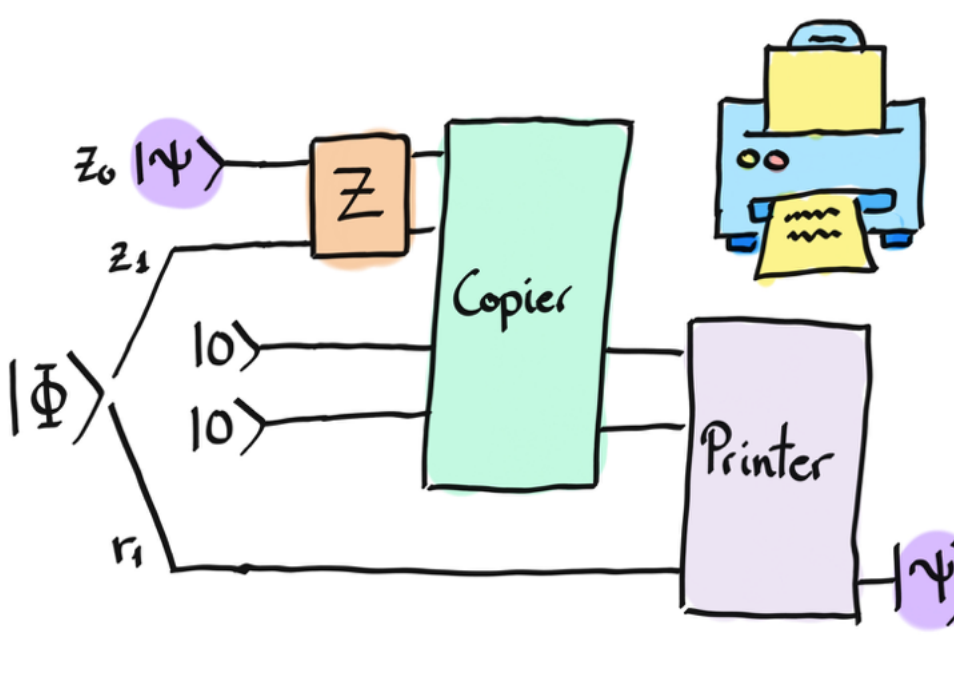
and extending linearly to the whole space. From linearity, you can prove yourself that the operator the photocopy machine is applying does not allow for copying arbitrary states!

Zenda ponders the meaning of Trine's words. She wonders if she can use the photocopy machine as a quantum fax machine instead. *That should be equivalent to quantum teleportation... except that I wouldn't need to do mid-circuit measurements*. She convinces Reece to join on the mischief, sharing with him half of the Bell state

$$|\Phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_{Z_1}|0\rangle_{R_1} + |1\rangle_{Z_1}|1\rangle_{R_1}).$$

Let's see how Zenda and Reece get away with this. Zenda has a state $|\psi\rangle$ that she wants to transfer to Reece, and half of the Bell pair above. After doing some operations Z on her two qubits, she can perform the `copier` operation that copies basis states into two registers inside the copy machine's server. That information is then transferred to Reece's printer where, after performing the `printer` operation with his states, he prints the state $|\psi\rangle$ into his half of the entangled pair.

Zenda shows Reece the schematics for the above:



Your task is to build the operator Z that Zenda must perform on her qubit, as well as the copier and printer operators needed to teleport the state. For the copier operator, the simplest way is to use the basis copying operator introduced at the beginning:

$$|j\rangle|0\rangle \mapsto |j\rangle|j\rangle,$$

Which well-known gate achieves this?

▼ Laws of Infodynamics Part IV: Coherent versions of the Third and Fourth Laws

This box contains some interesting but nonessential details.

The operation

$$|j\rangle|0\rangle \mapsto |j\rangle|j\rangle,$$

we introduced in this challenge is known as a **cobit**.

As a simple application of cobits, we can make a Bell pair by applying the photocopier to the $|+\rangle$ state:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

But cobits are much more interesting! As a first example, one can modify superdense coding so that Zenda and Reece can copy two bits:

$$|j, k\rangle_Z \mapsto |j, k\rangle_Z |j, k\rangle_R.$$

The basic idea is to replace the act of sending two classical bits, j and k , with the act of copying two basis states:

$$|j, k\rangle_Z \mapsto |j, k\rangle_Z |j, k\rangle_R.$$

We can rewrite the Third Law of Infodynamics (superdense coding) in terms of cobits as

$$1 \text{ qubit} + 1 \text{ ebit} \geq 2 \text{ cobits}, \quad (3')$$

where $x \geq y$ means having resource x also provide resource y , and "ebit" means "entangled bit", i.e. half a Bell pair. Similarly, it is possible to perform teleportation *coherently*, so that sending random bits j, k to correct the teleported state is instead a unitary involving $|j, k\rangle$. This leads to a new version of the Fourth Law of Infodynamics (teleportation):

$$1 \text{ ebit} + 2 \text{ cobits} \geq 1 \text{ qubit} + 2 \text{ ebits}. \quad (4')$$

If we subtract an ebit from both sides and combine with the modified Third Law, we obtain the *equality*

$$2 \text{ cobits} = 1 \text{ qubit} + 1 \text{ ebit}.$$

Photocopying basis states is precisely halfway between an ebit and a qubit!

Challenge code

In this challenge, you will be asked to complete the `zenda_operator`, `copier` and `printer` functions. All of them are quantum functions where you will only have to place the necessary gates.

Inputs

The inputs of this challenge correspond to the three coefficients of a $U3$ gate in charge of encoding the state $|\Phi\rangle$ that Zenda wants to send.

Outputs

To check the solution, we will calculate the expected value with respect to a particular observable to see that it coincides with the same one generated by Zenda. If your solution matches the correct one within the given tolerance specified in `check` (in this case it's a `1e-2` absolute error tolerance), the output will be "Correct!". Otherwise, you will receive a "Wrong answer" prompt.

Code

Help

```
1 import json
2 import pennylane as qml
3 import pennylane.numpy as np

4 def zenda_operator():
5     """
6     Quantum function corresponding to the operator to be applied by
7     Zenda on her qubits. This function does not return anything,
8     you must simply write the necessary gates.
9     """
10

11 # Put your code here #

13 def copier():
14     """
15     Quantum function encoding the copy operation done by Zenda, on each qubit.
16     This function does not return anything, you must simply write the necessary gates.
17     """
18

19 # Put your code here #

21 def printer():
22     """
23     Quantum function encoding the print operation done by Reece's printer.
24     This function does not return anything, you must simply write the necessary gates.
25     """
26

27 # Put your code here #

29 def bell_generator():
30     """
31     Quantum function preparing bell state shared by Reece and Zenda.
32     """
33
34     qml.Hadamard(wires=["z1"])
35     qml.CNOT(wires=["z1", "r1"])
36
37     dev = qml.device("default.qubit", wires=["z0", "z1", "r1", "s0", "s1"])
38
39     @qml.qnode(dev)
40     def circuit(alpha, beta, gamma):
41         # we encode the initial state
42         qml.U3(alpha, beta, gamma, wires = "z0")
43
44         bell_generator()
45
46         # Zenda acts on her qubits and establishes and copies them.
47         zenda_operator()
48         copier()
49
50         # Reece programs his printer
51         printer()
52
53         # Here we are returning the expected value with respect to any observable,
54         # the choice of observable is not important in this exercise.
55
56         return qml.expval(0.25 * qml.PauliX("r1") + qml.PauliY("r1"))
57
58
59
60

61 # These functions are responsible for testing the solution.
62 def run(test_case_input: str) -> str:
63     angles = json.loads(test_case_input)
64     output = circuit(*angles)
65     return str(output)
66
67
68 def check(solution_output: str, expected_output: str) -> None:
69
70     solution_output = json.loads(solution_output)
71     expected_output = json.loads(expected_output)
72     assert np.allclose(
73         solution_output, expected_output, atol=2e-1
74     ), "The expected output is not quite right."
75
76
77 try:
78     dev1 = qml.device("default.qubit", wires = ["z0", "z1"])
79     @qml.qnode(dev1)
80     def circuit1():
81         zenda_operator()
82         return qml.probs(dev1.wires)
83     circuit1()
84 except:
85     assert False, "zenda_operator can only act on z0 and z1 wires"
86
87
88 try:
89     dev1 = qml.device("default.qubit", wires = ["z0", "z1", "s0", "s1"])
90     @qml.qnode(dev1)
91     def circuit1():
92         copier()
93         return qml.probs(dev1.wires)
94     circuit1()
95 except:
96     assert False, "copy can only act on z0, z1, s0 and s1 wires"

107 test_cases = [['[1,1,1]', '0.8217355966267811'], ['[1.2,1.3,1.4]', '0.9604037313446201']]

108
109 for i, (input_, expected_output) in enumerate(test_cases):
110     print(f"Running test case {i} with input '{input_}'...")
111
112     try:
113         output = run(input_)
114
115     except Exception as exc:
116         print(f"Runtime Error. {exc}")
117
118     else:
119         if message := check(output, expected_output):
120             print(f"Wrong Answer. Have: '{output}', Want: '{expected_output}'")
121
122         else:
123             print("Correct!")
```