QHack

Quantum Coding Challenges

RANK	TEAM	CHALLENGES	SUBMISSIONS	SUPPORT

🚀 **CHALLENGE COMPLETED**   **View successful submissions**

⌄ **Jump to code**   — **Collapse text**

The Itch to Switch

500 points

Backstory

Zenda and Reece have been busy photocopying qubits and making their old communication protocols coherent. Zenda asks Trine what this has to do with timbits. Trine replies: "*Timbits? I forgot all about them. I suppose I wanted to show you there is more in heaven and earth than qubits and entangled pairs!*" Reece objects: "*But why did you get us to do all those protocols with photocopiers?*" Trine looks confused for a moment, then a smile spreads over her face. "*That's right! We can use them to implement a SWAP gate using two CNOTs as opposed to the usual three. Let's do that as a warm-up for timbits!*"

Exchanging qubits

Did you know that there is no way for us to clone a quantum state? The no-cloning theorem states that there is no gate U such that

$$U|\psi\rangle|0\rangle = |\psi\rangle|\psi\rangle$$

for all states $|\psi\rangle$. However, if we only work with basis states $|j\rangle$, there exist operations such that

$$|j\rangle|0\rangle \mapsto |j\rangle|j\rangle.$$

Zenda and Reece are each in possession of one basis state, which we denote $|j\rangle_{Z_0}$ and $|k\rangle_{R_0}$ respectively. Trine tells them to send each other their basis state to each other without losing their own. *"If basis states can be cloned, then surely we can do this"*, claims Zenda confidently. *"Just give us two qubits in the $|0\rangle$ state to each of us and we're good to go."*

Trine thinks about this... *"It's too easy if I allow you to do whatever you want"*—she concludes. *"Let's make it more fun. I'll give you each one qubit from a Bell state*

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

"Then you'll have to send your qubit to each other by acting only on the qubits in your possession."

Zenda and Reece try and try, but it seems like a futile task. *"We need more resources*—mumbles Reece. *"Mmm... disappointing"* says Trine. *"Then, I'll allow you to use a magic gate between your initially entangled qubits, but figure it out fast!"*

In this challenge, you will help Zenda and Reece figure out a quantum circuit that performs the operation

$$|j\rangle_{Z_0}|\Phi\rangle_{Z_1 R_1}|k\rangle_{R_0} \mapsto |j\rangle_{Z_0}|k\rangle_{Z_1}|j\rangle_{R_1}|k\rangle_{R_0}$$

with the constraints imposed by Trine. This means that the circuit must be of the form shown in the image below.
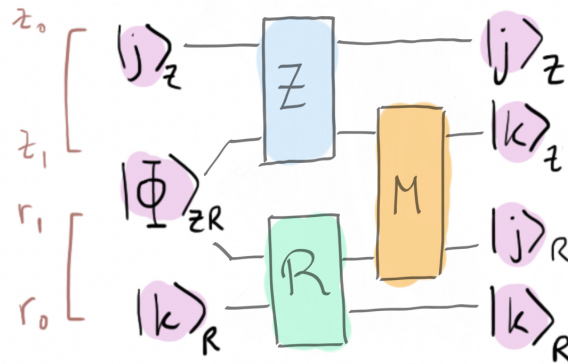
In the above, Z is the operator Zenda applies on her qubits, R is the operator Reece applies on his qubits, and M is the magic operator provided by Trine. This operation is one of the building blocks you need to master to build a SWAP gate with only two CNOTs, without counting the distributed CNOTs (contained in the magic gate). See the optional reading below for more about this!

▼ **Laws of Infodynamics Part V: From three to two CNOTs**

This box contains some interesting but nonessential details. As with previous Laws of Infodynamics, we can write the ability to perform one task in terms of others as an inequality:

$$1\,\text{CNOT} + 1\,\text{ebit} \geq 1\,\text{cobit}_{Z \to R} + 1\,\text{cobit}_{R \to Z}, \quad (\geq)$$

where the subscripts denote the source and target of copying. Similarly, it's possible to simulate a distributed CNOT gate from Zenda (control) to Reece (target) using a shared ebit and cobit exchange. This leads to an inequality

$$1\,\text{CNOT}_{Z \to R} + 1\,\text{ebit} \leq 1\,\text{cobit}_{Z \to R} + 1\,\text{cobit}_{R \to Z}. \quad (\leq)$$

We can combine the inequalities ($\leq$) and ($\geq$) into an *equality* for cobit swapping:

$$1\,\text{CNOT}_{Z \to R} + 1\,\text{ebit} = 1\,\text{cobit}_{Z \to R} + 1\,\text{cobit}_{R \to Z}. \quad (=)$$

This means that the resources on either side are equivalent! But this isn't just theoretical; it leads to a neat computational insight. We know that three

zigzag CNOTs are equivalent to a swap gate:

$$\text{SWAP}_{Z \leftrightarrow R} = \text{CNOT}_{Z \to R} \cdot \text{CNOT}_{R \to Z} \cdot \text{CNOT}_{Z \to R}.$$

We previously learnt that a cobit is the average of a qubit and an ebit:

$$2\,\text{cobit}_{Z \to R} = 1\,\text{qubit}_{Z \to R} + 1\,\text{ebit}_{Z \to R}.$$

If we double the equation ($=$) and use this average property, we find that

$$2\,\text{CNOTs} + 2\,\text{ebits} = 1\,\text{qubit}_{A \to B} + 1\,\text{qubit}_{B \to A} + 2\,\text{ebits}.$$

Subtracting two ebits from each side, the LHS is two distributed CNOTs (one from Reece to Zenda and vice-versa) and the RHS is a qubit sent in either direction. This is fancy terminology for a SWAP gate! Thus, we learn that only *two* CNOTs are needed to perform a SWAP:

$$2\,\text{CNOTs} = 1\,\text{SWAP}.$$

Pretty nifty huh! Note that we've subtracted four ebits in total from both sides, so we don't count the four CNOTs used to prepare these.

▼ **Epilogue**

Trine looks pleased. "*But all of this is merely an amusing diversion. Let's finally talk about timbits...*" At this very moment, Doc Trine, and the equipment she has been using to demonstrate the Laws of Infodynamics, disappear in a puff of sparkling purple smoke. Instead of lab equipment, there is only a business card, reading:

Ove A. Heard: Freelance Data Security Analyst
'We're Always Watching'

On the reverse, an elegant handwritten note:

Trine and the timbits are in hyperjail.
— Ove A. and OWT.

How can they rescue Doc Trine? Where is this hyperjail? And when will they learn the secret of timbits?!

Read on in **A Tale of Timbits.**

Challenge code

In the code below, you are given a number of functions:

- `zenda_operator` : Quantum function corresponding to the operator to be applied by Zenda on her qubits. **You must complete this function.**

- `reece_operator` : Quantum function corresponding to the operator to be applied by Reece on his qubits. **You must complete this function.**

- `magic_operator` : The magic operator provided by Trine to be applied on the initially entangled qubits Z_1 and R_1. **You must complete this function.**

Inputs and outputs

There are no inputs nor outputs for this challenge. You answer will be judged based on the fact that your circuit produces the correct final state for any combination of basis states $|j\rangle_{Z_0}$ and $|k\rangle_{R_0}$. This will be verified in the `check` function.

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
7       Zenda in her qubits.This function does not return anything,
8       you must simply write the necessary gates.
9       """
10

11       # Put your code here #
12

13   def reece_operator():
14       """
15       Quantum function corresponding to the operator to be applied
16       Reece in his qubits.This function does not return anything,
17       you must simply write the necessary gates.
18       """
19
```

```python
        # Put your code here #

    def magic_operator():
        """
        Quantum function corresponding to the operator to be applied
        and "r1" qubits. This function does not return anything, you
        simply write the necessary gates.

        """

        # Put your code here #

    def bell_generator():
        """
        Quantum function preparing bell state shared by Reece and Ze
        """

        qml.Hadamard(wires=["z1"])
        qml.CNOT(wires=["z1", "r1"])


    dev = qml.device("default.qubit", wires=["z0", "z1", "r1", "r0"]

    @qml.qnode(dev)
    def circuit(j, k):
        bell_generator()

        # j encoding and Zenda operation
        qml.BasisEmbedding([j], wires="z0")
        zenda_operator()

        # k encoding and Reece operation
        qml.BasisEmbedding([k], wires="r0")
        reece_operator()

        magic_operator()

        return qml.probs(wires=dev.wires)
```

```
59    # These functions are responsible for testing the solution.
60  v def run(test_case_input: str) -> str:
61        return None
62
63
64  v def check(solution_output: str, expected_output: str) -> None:
65
66  v     try:
67            dev1 = qml.device("default.qubit", wires = ["z0", "z1"])
68            @qml.qnode(dev1)
69  v         def circuit1():
70                zenda_operator()
71                return qml.probs(dev1.wires)
72            circuit1()
73  v     except:
74            assert False, "zenda_operator can only act on z0 and z1
75
76  v     try:
77            dev2 = qml.device("default.qubit", wires = ["r0", "r1"])
78            @qml.qnode(dev2)
79  v         def circuit2():
80                reece_operator()
81                return qml.probs(dev2.wires)
82            circuit2()
83  v     except:
84            assert False, "reece_operator can only act on r0 and r1
85  v     try:
86            dev3 = qml.device("default.qubit", wires = ["z1", "r1"])
87            @qml.qnode(dev3)
88  v         def circuit3():
89                magic_operator()
90                return qml.probs(dev3.wires)
91            circuit3()
92  v     except:
93            assert False, "magic_operator can only act on r1 and z1
94
95
96  v     for j in range(2):
97  v         for k in range(2):
98                assert np.isclose(circuit(j, k)[10 * j + 5 * k], 1),
99
100   test_cases = [['No input', 'No output']]
```

```python
101   for i, (input_, expected_output) in enumerate(test_cases):
102       print(f"Running test case {i} with input '{input_}'...")
103
104       try:
105           output = run(input_)
106
107       except Exception as exc:
108           print(f"Runtime Error. {exc}")
109
110       else:
111           if message := check(output, expected_output):
112               print(f"Wrong Answer. Have: '{output}'. Want: '{expe
113
114           else:
115               print("Correct!")
```

Copy all

Submit

Open Notebook ↗

Reset