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

⌄ Jump to code   — Collapse text

Sub-Superdense Coding

200 points

Backstory

Zenda and Reece have worked hard to implement classical computation reversibly. "*This is all rather boring though,*" says Trine. "*Let's introduce entanglement and have some fun!*" Trine tells them that the next Laws of Infodynamics will constrain how much classical information can be sent using entangled quantum resources.

Sending information with entangled states

Entanglement is a valuable resource that can be used to send information, as we will explore in this challenge. Superdense coding, for instance, uses Bell

pairs to send two classical bits with a single qubit.

Trine has misplaced her usual Bell states, and instead provides Zenda with 2 qubits and Reece with 1 qubit. They are entangled forming the following state:

$$|\Psi\rangle = \frac{|00\rangle_Z|0\rangle_R + |11\rangle_Z|1\rangle_R}{\sqrt{2}}.$$

Zenda wants to send Reece 3 bits of information (i, j, k), where each bit can take the value 0 or 1. To do so, she will apply an **encoding gate** on her two qubits and then send them to Bob.

Reece, who now has the three qubits and knows the strategy Zenda uses to encode, performs a decoding gate that will generate the state $|i, j, k\rangle$. Let's look at the following drawing to understand it.
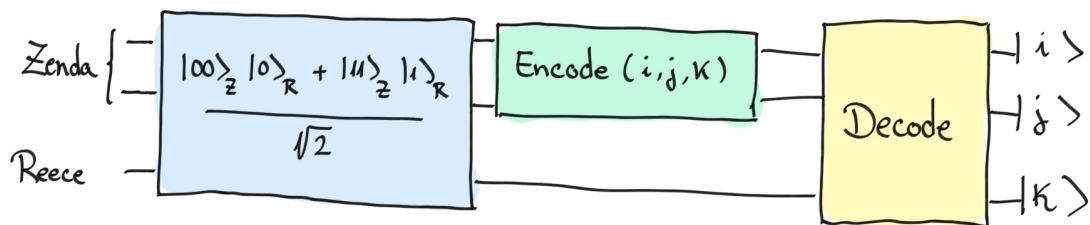
Your goal will be to devise a coding and decoding strategy such that Reece can decode Zenda's bits, as shown in the figure above.

The strategy to build the encoding gate is akin to superdense coding for two bits of information, where all the Bell basis states are produced by acting only on one of the qubits in an entangled pair. Here, we want to produce all the GHZ basis states, which are analogous to the Bell basis for three qubits. They are given by

$$|GHZ\rangle_{ijk} = \frac{1}{\sqrt{2}}\left(|0jk\rangle + (-1)^i|1\bar{j}\bar{k}\rangle\right),$$

where $\bar{j} = 1 - j$ and $\bar{k} = 1 - k$. However, we should do this by acting only on the first two qubits (i.e. Zenda's share) of the entangled state $|\Psi\rangle$ that Trine provided. The decoding gate is also inspired by superdense coding, so reviewing the topic will help you a lot in this challenge!

Challenge code

You simply have to complete two quantum functions:

- `encode` : quantum function that will define an operator to be applied only on Zenda's qubits. This function will depend on the bits (i, j, k) to be encoded.

- `decode` : quantum function that defines the operator that Reece will use to retrieve the bits that Zenda sent. In this case, the operator does not have any information about Zenda's bits, so the same operators will always be applied regardless of the state that Zenda sends Reece.

Output

In this challenge, we will not judge your solution using public or private test cases. Instead, we will check that, for all combinations of i, j, and k, the entire encoding and decoding circuit behaves as expected.

Code   ? Help

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

3   def encode(i, j, k):
4       """
5       Quantum encoding function. It must act only on the first two
6       This function does not return anything, it simply applies ga
7
8       Args:
9           i, j, k (int): The three encoding bits. They will take t
10
11      """
12
```

```
13      # Put your code here #
14
```

```python
15 v  def decode():
16         """
17         Quantum decoding function. It can act on the three qubits.
18         This function does not return anything, it simply applies ga
19         """
20

21         # Put your code here #
22

23    dev = qml.device("default.qubit", wires=3)
24
25    @qml.qnode(dev)
26 v  def circuit(i, j, k):
27         """
28         Circuit that generates the complete communication protocol.
29
30         Args:
31             i, j, k (int): The three encoding bits. They will take t
32         """
33
34         # We prepare the state 1/sqrt(2)(|000> + |111>)
35         qml.Hadamard(wires=0)
36         qml.CNOT(wires=[0, 1])
37         qml.CNOT(wires=[0, 2])
38
39         # Zenda encodes the bits
40         encode(i, j, k)
41
42         # Reece decode the information
43         decode()
44
45         return qml.probs(wires=range(3))
46
```

```python
47    # These functions are responsible for testing the solution.
48
49
50
51 v  def run(test_case_input: str) -> str:
52
53        return None
54
55 v  def check(solution_output: str, expected_output: str) -> None:
56
57 v       for i in range(2):
58 v           for j in range(2):
59 v               for k in range(2):
60                      assert np.isclose(circuit(i, j , k)[4 * i + 2 *
61
62                      dev = qml.device("default.qubit", wires=3)
63
64                      @qml.qnode(dev)
65 v                    def circuit2(i, j, k):
66                          encode(i, j, k)
67                          return qml.probs(wires=range(3))
68
69                      circuit2(i, j, k)
70                      ops = circuit2.tape.operations
71
72 v                    for op in ops:
73                          assert not (2 in op.wires), "Invalid connect
74
75    test_cases = [['No input', 'No output']]

76 v  for i, (input_, expected_output) in enumerate(test_cases):
77        print(f"Running test case {i} with input '{input_}'...")
78
79 v       try:
80            output = run(input_)
81
82 v       except Exception as exc:
83            print(f"Runtime Error. {exc}")
84
85 v       else:
86 v           if message := check(output, expected_output):
87                print(f"Wrong Answer. Have: '{output}'. Want: '{expe
88
89 v           else:
90                print("Correct!")
```

Copy all

Submit

Open Notebook ⬀

Reset