PENNY GAMES

Games: Tardigrade Masquerade [100 points]

Version: 1

Games

Quantum computers help us solve many problems, but we can also use them to explore some of the most mysterious aspects of quantum mechanics. We learn so much better when we have fun, so why not use some crazy games and experiments to explore the boundaries of quantum theory? These fun coding challenges, ranging from entangling full-blown animals to using quantum circuits to cheat our way into victory, teach us a lot about why quantum computing is so powerful.

In the **Games** category, we will be exploring some of the weirdest quantum experiments proposed in the literature. These will enlighten us about how quantum mechanics is different from classical physics, but will also give rise to deeper philosophical questions. Let's have some fun!

Problem statement [100 points]

In this paper by K.S. Lee *et al.* they claimed to have demonstrated entanglement between superconducting qubits and a tardigrade (WARNING: it's ugly).

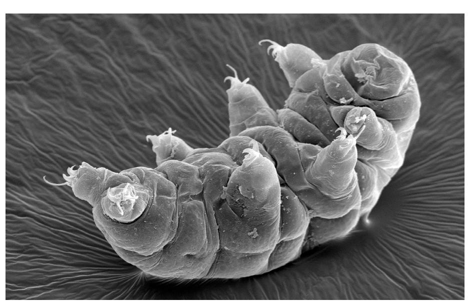

Figure 1: The tardigrade

In this challenge, you will replicate some of their results. The idea is to prepare two states: one with a tardigrade and one without. After state preparation, we will calculate a measure of entanglement on both states and compare.

Firstly, you must prepare the following two-qubit state:

$$|\phi\rangle_{AB} = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B \right).$$

Here, there is no tardigrade present and A and B are labels for two qubits. Next, you will create the following three-qubit state:

$$|\psi\rangle_{ABT} = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |e\rangle_{BT} + |1\rangle_A \otimes |g\rangle_{BT} \right),$$

where

$$|e\rangle_{BT} = \cos\tfrac{\theta}{2} |1\rangle_B \otimes |0\rangle_T + \sin\tfrac{\theta}{2} |0\rangle_B \otimes |1\rangle_T,$$
$$|g\rangle_{BT} = |0\rangle_B \otimes |0\rangle_T,$$

and T labels the qubit that represents the tardigrade. The qubits A, B, and T should be represented in your code as qubits/wires 0, 1, and 2, respectively.

To demonstrate the existence of entanglement introduced by the presence of the tardigrade, we will calculate a well-known measure of entanglement entropy called the second Rényi entropy with respect to qubit B,

$$S_2(\rho_B) = -\ln \mathrm{Tr}(\rho_B^2),$$

where the reduced state ρ_B is obtained by tracing out all other subsystems. For example, in the tardigrade state ρ_{ABT}, we trace over the degrees of freedom associated with subsystem A and T:

$$\rho_B = \mathrm{Tr}_{A,T}(\rho_{ABT}),$$

where

$$\rho_{ABT} = |\psi\rangle\langle\psi|_{ABT}.$$

Notice that if the reduced state ρ_B is maximally mixed, i.e., $\rho_B = \frac{1}{2} I$, then its second Rényi entropy is

$$S_{\rho_{B,\text{max. mixed}}} = -\ln \mathrm{Tr}\left(\frac{1}{4} I\right) = -\ln \frac{1}{2} \approx 0.693.$$

This is the maximum value of the second Rényi entropy, meaning that we have the most entanglement in this case. This is actually the case for the "tardigrade-less" state

$$\mu_{AB} = |\phi\rangle\langle\phi|_{AB},$$

i.e., $\mu_B = \mathrm{Tr}_A(\mu_{AB})$ here is maximally mixed, meaning μ_{AB} is maximally entangled. You need to determine what happens to the entanglement entropy when the tardigrade is introduced. Specifically, calculate the second Rényi entropy of the reduced states ρ_B and μ_B.

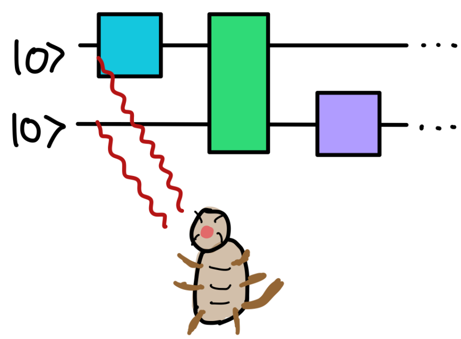

Figure 2: Tardigrade Masquerade

The template file `tardigrade_template.py` contains two functions. The `second_renyi_entropy` function will calculate the second Rényi entropy of a given density matrix. The `compute_entanglement` function will be where you need to prepare the two states given above (see `mu_B` and `rho_B`). The quantum functions you use to prepare both states need to output the reduced density matrix describing only qubit B. With these two density matrices in hand, calculate their second Rényi entropies using the `second_renyi_entropy` function.

Input

- `float`: The angle θ.

Output

- `list(float)`: Rényi Entropy of the tardigrade-less state, Rényi entropy of the state with the tardigrade present.

Acceptance Criteria

In order for your submission to be judged as "correct":

- The outputs generated by your solution when run with a given `.in` file must match those in the corresponding `.ans` file to within the `0.0001` tolerance specified below. To clarify, your solution must satisfy

$$\text{tolerance} \geq \left| \frac{\text{your solution} - \text{correct answer}}{\text{correct answer}} \right|.$$

- Your solution must take no longer than the `60s` specified below to produce its outputs.

You can test your solution by passing the `#.in` input data to your program as stdin and comparing the output to the corresponding `#.ans` file:

```
python3 {name_of_file}.py < 1.in
```

WARNING: Don't modify the code outside of the `# QHACK #` markers in the template file, as this code is needed to test your solution. Do not add any print statements to your solution, as this will cause your submission to fail.

Specs
Tolerance: **0.0001**
Time limit: **60 s**

Version History

Version 1: Initial document.