Programming a quantum computer – Day 3
Exercises

Peter Brown

June 22, 2022

I've prepared some notes on Grover's algorithm, see the notes folder in the github repository we've been using. I would recommend that you go through these notes and try to understand this algorithm together (it's a nice one!). You can also use other references to learn Grover's algorithm like the Nielsen and Chuang book or Ronald de wolf's lecture notes which can both be found online. There's some questions prepared based on this algorithm to help you solidify your understanding.

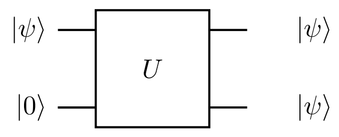
1 Theory questions

1. **Classical database search**

   Suppose you are searching a database of N items for a specific entry. Show that classically you need on expectation $N/2$ queries to find the item you are looking for.

2. **No cloning**

   In quantum theory there is a concept called *no cloning* which states that there is no single unitary U such that



(1)

   for every state $|\psi\rangle$. I.e., you cannot have a *universal* quantum copier and you cannot copy unknown quantum states.

   (a) Prove that no unitary U exists such that

   $$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \qquad \text{for all } |\psi\rangle . \tag{2}$$

   (b) Why does quantum teleportation not violate the no-cloning principle?

   (c) No-cloning feels like a limitation of quantum theory at first. Can you think of advantages that come from no-cloning?

3. **Grover's algorithm (based on notes)**

   I will use the same notation as was used in the notes on Grover's algorithm.

   (a) We can write

   $$|+\rangle^{\otimes n} = \sin(\theta)\,|\mathbf{w}\rangle + \cos(\theta)\,|Bad\rangle . \tag{3}$$

   but what is the value of θ?

   (b) Show that the action of the oracle gate $U_{\mathbf{w}}$ can be interpreted in the diagram as a reflection about the vector $|\mathrm{Bad}\rangle$.

(c) Show that the action of the S gate can be interpreted in the diagram as a reflection about the vector $|+\rangle^{\otimes n}$.

(d) How many times should we apply the gate G to maximize our probability of getting the correct solution? Show that this is $O(\sqrt{N})$.

(e) Using the geometric intuition, what would appear to happen if we applied the Grover gate G too many times?

(f) (More difficult) Can you extend Grover's algorithm to deal with the case where there are M different entries in the database such that $f(\mathbf{x}) = 1$? (But you only care about finding any one of them – not all of them.)

2 Practical exercises

Today we'll implement some simple quantum algorithms and look at error correction.

1. **Deutsch-Josza algorithm**

   We saw in class the Deutsch-Josza algorithm as a method to distinguish between a constant or balanced Boolean function. You will now be given access to some quantum oracles that implement such a function and you are tasked for each oracle to check whether it is constant or balanced. These oracles can be accessed on the qiskit textbook github repository, you will need to install the qiskit textbook module

```
pip install git+https://github.com/qiskit-community/qiskit-textbook.git#subdirectory=qiskit-textbook-src

```

   you can then access the oracles via

```
from qiskit import QuantumCircuit, QuantumRegister
from qiskit_textbook.problems import dj_problem_oracle

# Select the oracle to test
ORACLE_NUMBER = 1
oracle = dj_problem_oracle(ORACLE_NUMBER)

# Initialize the circuit
qr = QuantumRegister(5, name="q")
qc = QuantumCircuit(qr)

# Method to apply the oracle
qc.append(oracle, [0,1,2,3,4])

```

   Qubits 0-3 are the input qubits and qubit 4 is the function output qubit updated to $|f(x)\rangle$, there are 5 different oracles to try labelled (1,2,3,4,5).

   (a) Design and implement a circuit to decide whether the oracles (1-5) are implementing a constant or a balanced function using a single query. IBM recommends using the simulator for this task but it may also be possible to implement on one of the machines.

2. **Grover's algorithm**

   (a) Work through an example of Grover's algorithm for a small number of qubits (less than 5) and try simulating and implementing it on the IBM quantum computers! Note that this will involve you having to make up your own oracles. Like in the previous day you can design an oracle for a friend to use by creating a function grover_oracle() that applies the oracle gate to their circuit. They can then try to run the algorithm to see if they can find the database entry you selected.

   (b) (Extension) If you were able to extend Grover's algorithm to more success bitstrings then can you also run this instance on the machines?

3. **Quantum error correction**

   (a) Try implementing the bitflip and phaseflip codes seen in class (see notes) on the IBM machines (5 qubits should be sufficient for these codes for the full encoding/detection/correction/decoding). Check that they work on X and Z errors on the different qubits respectively.

   (b) What happens if you make other errors with these codes, do they fail?

   (c) (For the brave) Unfortunately, with only 5 qubits we cannot do full quantum error correction.[1] However, we can still write code to simulate quantum error correction. Try writing the full Shor 9-qubit code to perform full error correction for a single qubit error. This will involve writing the encoding/decoding circuits as well as the error-detection and correction steps.

4. **Further exploration**

   (a) There are many other instances of quantum algorithms out there (some more complicated than others). For a curated list, check out the quantum algorithm zoo. You are free to explore these and try to understand any that seem interesting to you. You can try presenting one to other members of your group!

   For some reference, QISKIT's textbook has a guide for the following algorithms (non-exhaustive): Bernstein-Vazirani, Simon's, Shor's (a very well known algorithm for its impact on cryptography), HHL and many more...

   (b) There is also a plethora of quantum error correcting codes. So many that it also comes with a dedicated website, the error correction zoo. This is probably more difficult to digest than the algorithms but if you're interested you can still take a look.

[1] It is possible to enquire to IBM about accessing the larger machines but I've not tried this.