Physics 4481-7681; CS 4812; AEP 4812-7681 ps5 part 2

Four additional problems, covers lectures 17–19, due Fri evening 11 Nov 2022 at 23:00

Problem 1: See Jupyter notebook ps5.prob1.ipynb

Problem 2: Toffoli gates

a) Consider the $T$ gate with $T^4 = Z$ (as considered in ps2#4), which together with $H$ and cNOTs provides a universal gate set.*

\[
\begin{array}{c}
\oplus \quad T \\
\oplus \quad T^\dagger \\
\oplus \quad T \\
\end{array}
= \quad \begin{array}{c}
\oplus \quad S^\dagger \\
\oplus \quad -iZ \\
\end{array}
\]

Verify the circuit identities above, showing that the circuit at left realizes an operator closely related to ‘CCZ’ (double-controlled $Z$ operator, in turn easily converted to a Toffoli gate using a pair of $H$ gates around a target).† As usual, the operator $S = T^2 = \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$. Rather than trying to run the 8 possible basis states through the gates, try to establish it algebraically in terms of the phase $\omega = e^{i\pi/4}$. For example with inputs $|x\rangle, |y\rangle, |z\rangle$ at left, the middle qubit after the first cNOT is in state $z \oplus y$, and the upper qubit after the 2nd cNOT is in state $z \oplus y \oplus x$. The phase contributed by the upper left $T$ gate is therefore $\omega^{x \oplus y \oplus z}$. By similar reasoning, it’s possible to multiply by phases contributed by the other three $T, T^\dagger, T^\dagger$ gates. Then via repeated use of $x \oplus y = x + y - 2xy$, it’s possible to derive a simple identity for $(-1)^{xyz}$ to show that the resulting phase is algebraically equivalent to either of the circuits on the right.

b) Using the same method, verify that this circuit implements the Toffoli gate:

\[
\begin{array}{c}
\oplus \\
\oplus \\
\oplus \\
\end{array}
= \quad \begin{array}{c}
\oplus \quad T \\
\oplus \quad T^\dagger \\
\oplus \quad T \\
\end{array}
\]

(where again instead of considering basis states, keep track of powers of $\omega$, and use the same identity for $(-1)^{xyz}$ as in part a)).

[Note: in ps5#1a you need to create a Toffoli gate, and the version in 2b above is the one mentioned at https://qiskit.org/textbook/ch-gates/more-circuit-identities.html#4.-The-Toffoli-]

* i.e., combinations of which can be used to realize any multi-qubit unitary transformations.

† The ‘CCZ factory’ plays a large role in anticipated realizations of Shor’s algorithm, providing the Toffoli gates necessary for implementation of the modular exponent.
Problem 3:

In lec18 (see slides 2,3 via Ed), it’s shown how to efficiently calculate the periodic function \( f(x) = b^x \mod N \), highlighting that it can be done with a single execution of the procedure that does the multiple squarings (without need of a classical look-up table).

(i) Recall that the time for the period finding is dominated by the calculation of the modular exponent function itself (as also discussed in lecture 18). Considering the number of modular multiplications, and how many additions each involves, and how many operations are involved in each addition, explain why calculating the modular exponent is \( O(n^3) \) (where \( n \) is the number of digits).

ii) Given the above estimate of how the number of gates scales in the number of digits of the number to be factored, roughly how many total gates (modular exponentiation plus quantum Fourier transform) might be necessary to factor the 2048 bit RSA number (the largest).
Note: the details will depend on the implementation, this is just to estimate some rough number.

(iii) Use the argument given in the right hand column of p.2 of arXiv:1208.0928 (and using the 100ns measurement time) to estimate how long it would take to factor the 829 and 2048 bit (250 and 617 decimal digit) numbers in ps4#2A, if the quantum method were currently feasible.
(The point here is just to confirm that \( n^3 \) grows much more slowly than \( \exp(an^{1/3}) \).
The method in the article results in an estimate of \( 40n^3 \) for the scaling of the number of operations to calculate the modular exponent function.)

(iv) [Bonus] arXiv:1905.09749 was also mentioned in class (Lec 14,18), and finds the period of a slightly different function than Shor’s \( b^x \mod N \) (see sec 2.A. of the article, using classical number theory). This method still permits factoring \( N \), but with fewer overall multiplications, reducing the coefficient of the \( n^3 \) term in part (iii) to \( .3n^3 \) (see Table I in the article). What is the effect on the results of part (iii) for the two values of \( N \)?
(Feel free to consider or ignore other optimizations discussed in the article.)
**Problem 4**: Searching for one of four items

The end of lec 19 (and M 4.5) treated the case of Grover’s algorithm applied to identify one of only four items. This was a special case in two ways: (1) The probability of success after just a single query of the oracle was exactly 1. (2) The general procedure for constructing \( W = 2\phi\langle\phi| - 1 \) out of Toffoli gates did not apply, and a different (simpler) construction could be used (though it was described slightly incorrectly in M 4.5).

a) Write down circuit diagrams (each in terms of a single Toffoli gate) for the four possibilities for the function \( U_f \) in this case, where the top two wires carry the the input register, and the bottom wire is the output register which is flipped if and only if the top two wires represent the special number. In the usual convention in which the most significant bit is carried by the top wire, identify which of the four possibilities is associated with the special number being 00, 01, 10, and 11.

b) Write a circuit diagram that acts as \( 1 - 2|11\rangle\langle11| \) (it needs only two wires for the two Qbits).

c) Write down the full circuit diagram to produce a single Grover iteration applied to the input state \(|0\rangle|0\rangle|1\rangle\). It should include a 3-Qbit gate as a box labelled \( U_f \) to represent any of the four possibilities of part (a).

d) For one of the four possibilities in part (a), explain why the final state of the input register is exactly the state marked as special. (It is probably simplest to calculate directly what the rest of the circuit does to the state for the case when the special input number is 11, though the other cases are easily related to that case.)

**Problem 5**: Modified Grover

a) In the notation used in lecture 19, with \( |\phi\rangle = \cos \theta |\text{no}\rangle + \sin \theta |\text{yes}\rangle \) (where assuming that the search problem has \( M \) solutions \( M \neq 0, M \neq N = 2^n \)), define \( |\text{yes}\rangle = \frac{1}{\sqrt{M}} \sum_x |f(x) = 1 \rangle \), and \( |\text{no}\rangle = \frac{1}{(2^n - M)^{1/2}} \sum_x |f(x) = 0 \rangle \), show that a single Grover iteration can be used to find a marked state whenever \( M/N = 1/4 \).

b) Consider a modification to Grover’s algorithm, where the ‘oracle’ now performs \( V_\alpha |x\rangle = e^{i\alpha} |x\rangle \), if \( |x\rangle \) is a target state, and \( V_\alpha |x\rangle = |x\rangle \) otherwise. In addition, take \( W_\alpha = (1 - e^{i\alpha})|\phi\rangle\langle\phi| - 1 = H^\otimes n \left((1 - e^{i\alpha})|0\rangle\langle0| - 1\right)H^\otimes n \).

Show that if you use \( G_\alpha = W_\alpha V_\alpha \) instead of the standard Grover iteration (note that \( \alpha = \pi \) corresponds to the usual \( G = WV \)), you can choose \( \alpha \) so that the algorithm finds a target state with probability 1 after one iteration, for any state with \( M/N \geq 1/4 \).
(Hint: find the coefficient of |no\rangle in the |yes\rangle, |no\rangle expansion of \( G_\alpha |\phi\rangle \) and set it to zero.)

c) Combine the result of b) with the standard Grover iteration \( \alpha = \pi \) to find a modified Grover’s algorithm that finds a marked state with probability 1 with the smallest number of ‘oracle’ calls, for any given \( M/N \).

In particular, for \( M = 1 \), what is the smallest number of calls to \( V = V_\pi \) (as part of \( G_\pi \)) combined with a single \( V_\alpha \) (as part of \( G_\alpha \)) to find the marked state with probability 1?