CSE491/596, Fall 2020 Problem Set 7 Due Fri. Dec. 11, 11:59pm

Plus topics for presentations on Dec. 10–11

Lectures and Reading: The remaining lectures will finish the coverage of quantum computing, with emphasis on chapters 7–9 (and possibly 10) of the Lipton-Regan text. (Chapters 11–13 are covered in CSE696.) On the way there, section 4.5 (which furnishes some examples for possible use in presentations), sections 5.1–5.3 (a case of the solved problem is another presentation option), and sections 6.1–6.4 are the ones to focus on.

Assignment 7, written part due Fri. Dec. 11, 11:59pm

(1) (12 + 18 = 30 pts.)

Let us revisit problem (2b) on Prelim II. The first part shows how reducing from Exactly One 3SAT shows the completeness of a more restrictive version of the problem. The second part challenges you to get the same conclusion by a reduction from ordinary 3SAT. Recall that a "star" in a (possibly directed) graph G to be a node u together with all nodes v such that u has an edge to v. The target problem in both cases is:

Exact Star Cover

Instance: A directed graph $G = (V, E)$.

Question: Is there a set of stars that includes every node in V exactly once?

(a) Explain why the answer-key solution also constitutes a mapping reduction from Exactly One 3SAT to Exact Star Cover. You must give the correctness part formally, using implications $\implies$. Prove in your answer why this works even though "k" has been taken out of the problem altogether.

(b) Now observe that this doesn't work as a reduction from 3SAT to Exact Star Cover because solutions that satisfy two or three literals in a clause include the node for that clause more than once. A similar problem was given (as makeup) last year and included at the end of the regular key:

https://cse.buffalo.edu/~regan/cse491596/CSE596F19ps4key.pdf

That problem was called "Perfect Dominating Set and included the k. Your question is: does the same construction work for the Exact Star Cover problem stated without k? Again, you must prove your answer with $\implies$ logic if you say less, or sketch a counterexample if you say no.

(2) (12 pts.)

Show that if TQBF belongs to NP then $\mathsf{PSPACE} = \mathsf{NP} \cap \mathsf{co\text{-}NP}$. [This is stated on page 2 of the notes for Nov. 16 but a few more facts than what is given need to be strung together to make a proof.]

(3) (18 pts. total)

Here is a version of the game "Chutes and Ladders" (also called "Snakes and Ladders") that you play by flipping a fair coin instead of rolling dice. The game board has cells $1, \dots, n$ where you start on 1 and n is the goal. Each cell is connected to the next cell but may also have a *ladder* going to a higher cell and/or a *chute* (or snake) going to a lower-numbered cell.

The rules with a coin are that if you are on cell m and get heads then you climb a ladder if there is one, else you advance to cell $m+1$. If you get tails, then you must go down a chute if there is one, else you go back to cell $m-1$ (or stay at the start if $m = 1$). We will play the game as solitaire, so the object is to tell how long you expect to need to play before you win.

First, suppose the game board is *undirected* in the sense that every chute from m to k is accompanied by a ladder going from k to m. (An equivalent way of saying this is that the game board has only ladders, but if you are at the top of a ladder and get tails, then you have to go down it.) Show that the expected time to win is $O(n^2)$. A fact you can quote is that the standard deviation of the binomial distribution counting heads from N coin flips is $0.5\sqrt{N}$.

Give an example, however, of fiendish chutes-only game boards for which you can show that the expected time to win is exponential in n.

[*Footnote:* To get the meaning of this, consider any undirected n-node graph G. If G has a path that starts at some node s and goes through all the nodes ending at some node t, then we can number the nodes in the graph 1 to n along that path. All other edges in the graph become ladders. If G does not have such a path (called a Hamiltonian path—whether an undirected graph has one is an NP-complete problem mentioned in Debray's notes but that lectures didn't cover), it doesn't matter much to the argument: Given any goal node t, let s be some node furthest from t. Every node has at least one option that gets closer to t, and that's all you need to define the game for G. These rules are not quite the same as the way a *random walk* in G is usually defined, but the effect is equivalent: the undirected version UGAP of the GAP problem (is there a path from s to t? and if so, find one) belongs to the logspace analogue SRL (symmetric random logspace, also called just SL) of RP. But in a *directed* graph, the argument trying to do the original GAP problem in randomized logspace *fails.*]

(4) (12 pts. total)

Lipton-Regan text, chapter 4, problems 4.10–4.12. (Note that they refer to problem 4.8, which along with 4.9 is solved in the selected answers section.)

(5) (18 pts. total)

Consider the **H**-**T**-**H** circuit at the beginning of section 4.5 of the Lipton-Regan text (2nd. ed. only). Add a second qubit line and consider inserting one more gate, a CNOT. There are 4 possible ways to place it: before the first Hadamard, between the first Hadamard and the **T**-gate, right after the **T**-gate, or at the end. Do all four preserve the property that on input e_{00} (that is, input $|00\rangle$), the probability of measuring 0 on line 1 is irrational? Show two relevant calculations using matrices by hand. (You can argue symmetry for the other two. This makes 90 points on the written part.)

Presentation Options For 12/10–11, all teamable

Each presentation option involves choosing a moderately-sized quantum circuit and demo'ing it on one of various quantum circuit applets that are freely available (some upon registration). Show how to build the circuit in the applet (if you save it in advance, please leave a few gate-insertion steps to do by hand), describe what it is intended to do, and show its behavior on a few relevant states.

This session is intended to be more about participation than presentation and to involve playing around with the circuits by adding, removing, or changing some gates. So please come prepared not only to give a riff or two on your example but also on the others. You don't need to know how to operate *all* the possible applets to give suggestions, just your own. Here are some applets:

1. Davy Wybiral's Quantum Circuit Simulator: https://wybiral.github.io/quantum/ (general page https://wybiral.github.io/) Most lightweight and user-friendly, IMHO, but somewhat limited.
2. Quirk: https://algassert.com/quirk, by Craig Gidney, a co-author of Google's 2019 quantum-supremacy paper. (He has lots of stuff at https://algassert.com/about.html too.)
3. The IBM Quantum Experience: https://quantum-computing.ibm.com/. Allows you to actually execute small circuits on a remote quantum machine. See also Qiskit (https://qiskit.org/textbook/ch-algorithms/defining-quantum-circuits.html)
4. Quantum Programming Studio: https://quantum-circuit.com/home Also has links to hardware-run options.
5. Q.js: https://quantumjavascript.app/ Seems the most limited.

There is an enormous list of simulators at https://www.quantiki.org/wiki/list-qc-simulators Now here is a list of possible circuits to demo. The last two are more open-ended. Other ideas of similar size can be cleared with me:

(a) Various *graph-state* circuits as described (briefly) in section 4.5 of the Lipton-Regan text (2nd ed. only). Build one(s) of 5 and 6 vertices/qubits too, such as the prism graph in section 3.6. One thing to look for is whether 0^n is given as a possible output when 0^n is the input. The significance of this is discussed in my article https://rjlipton.wordpress.com/2019/06/10/net-zero-graphs/ with my recent PhD graduate Chaowen Guan.

(b) The circuit for the Toffoli gate on the next page in section 4.5.

(c) The circuit to build the 4-qubit quantum Fourier transform from smaller pieces in the answer section of chapter 5. (Or work out the smaller 3-qubit version, which would be quite enough.)

(d) The circuit for quantum teleportation in section 8.3 of the text—where you can prepare an interesting single-qubit c by doing **HTH**, for instance.

(e) Some 3-qubit cases of the Deutsch-Jozsa algorithm in chapter 9, where you drop in gates for at least three examples of a binary function f.

(f) Some illustrations using $3 + 1 = 4$ or $3 + 2 = 5$ qubits of the copy-uncompute trick in section 6.3. Use at least one gate **A** with both a real and a complex number entry, such as **S** or **T**, so that its adjoint $\mathbf{A}^*$ is really different. The idea is to show that when the 1 or 2 intended outputs on the first 3 lines are basis values, then they get copied, but if they are still in superposition, "YMMV"—so this doesn't violate the no-cloning theorem. Using more qubits and complex gates might give an example that is more visually convincing than the one in the text.