CSE439, Fall 2026 Problem Set 6 Due Fri. 11/13, 11:59pm

The **Second Prelim Exam** will be on **Thursday, Nov. 19** *in class period*. It will cover up through section 14.4. For next week, please read chapter 14 up through section 14.7 and also chapter 18, section 18.1, only up through Theorem 18.1 and its proof.

————————*-Assignment 6, due Fri. 11/13 "midnight stretchy" on CSE Autolab*————————-

(1) Exercise 13.7 in chapter 13. An important point is that the Grover oracle $\mathbf{U}_G$ for a particular graph G is given separately and has all the needed information about G. The oracle is also *compact* in the following sense: it uses only ℓ qubits, where $L = 2^\ell$ is the next power of 2 above the number of potential solutions. Here the number of potential solutions is the binomial coefficient $\binom{n}{3}$ (which $\sim \frac{1}{6}n^3$), since any trio of nodes could be a triangle in G.

Explain how using just $\mathbf{U}_G$ on ℓ qubits, you can find a triangle (provided one exists) with high probability in the time specified. The word "about" means the time can be technically $\tilde{O}(n^{3/2})$, where the tilde means that factors of $\log n$ are ignored. You may assume that $\mathbf{U}_G$ is already coded via quantum gates (Toffoli gates are good enough since they can simulate NAND) and can ignore how many gates or how much time it took to build it—you can treat it as a "black box." So you need not fuss over details of $\mathbf{U}_G$ and may suppose that whatever the set S of actual triangles in G, $\mathbf{U}_G$ gives you reflection around the "miss vector" $\mathbf{m}_S$. Much of this is just digesting key details of Chapter 13—including a brief recap of section 13.3 since you don't know $|S|$—with the point being how and why it works out to time $\tilde{O}(n^{3/2})$. (18 pts. total)

[For some further chitchat: A classical algorithm has to be given G not $\mathbf{U}_G$. It can brute-force search for triangles in time $\tilde{O}(n^3)$ by trying each trio of nodes. There is another way by taking the $n \times n$ adjacency matrix A of G and computing its "Boolean square" $A^{[2]}$ under the rule that any sum greater than 1 counts as 1. Then G has a triangle if and only if $A + A^{[2]}$ has an entry 2 somewhere. This can be done in $n \times n$ matrix multiplication time, which AI recently improved to $\tilde{O}(n^{2.25})$. Whether this needs only time $\tilde{O}(n^2)$—which is linear in the max number $N = n^2$ of edges in the graph—is a major open question in complexity theory. Thus this problem tries to say Grover's algorithm can save time at polynomial-in-n rather than exponential-in-n scales. But this still ignores the time to create $\mathbf{U}_G$ to begin with.]

(2) Lipton-Regan text, problem 14.4 in chapter 14. This should include a check that $\theta = \frac{\pi}{4}$, $\varphi = 0$ in Bloch sphere coordinates is the same one-qubit state vector as $[\cos(\frac{\pi}{8}), \sin(\frac{\pi}{8})]$ in Cartesian coordinates. (18 pts. total)

(3) Lipton-Regan text, exercise 14.7 on page 165: Use the spectral method to compute a square root of

$$\mathbf{W} = \frac{1}{\sqrt{2}}(\mathbf{X} + \mathbf{Y}) = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 1-i \\ 1+i & 0 \end{bmatrix}.$$

(Please show all scratchwork. 18 pts., for 54 on the set)