A. You learned about graphs in CS 201. A *triangle* in a graph is a set of three vertices such that all three possible edges among those vertices are present. Consider the problem of testing whether a graph of m vertices contains a triangle.

In what follows, you can assume that the graph is stored in a very fast data structure. The vertices are numbered $0, 1, \ldots, m-1$, and you can ask whether there is an edge between vertex j and vertex k in constant time.

1. Devise a classical algorithm for solving this problem. What is its time complexity?

2. Devise a quantum algorithm based on Grover's algorithm. (If you like, you can use your unknown $k \geq 0$ algorithm from problem B below.) What is its time complexity? Be detailed.

(This is a medium-length problem.)

B. We have sketched a version of Grover's algorithm for unknown k, that works as long as $k \geq 1$. How could you adapt it into a version that works for unknown k that works as long as $k \geq 0$? Please solve this problem in two ways.

1. Solve the problem by modifying f into a new classical function g before making the corresponding gate G for use in Grover's circuit.

2. Solve the problem by using our algorithm for unknown $k \geq 1$ without modification, but then modifying how you interpret the outputs produced by that algorithm.

(This is a medium-length problem.)

C. (This problem is optional. Do not hand it in.) Consider the version of Grover's problem/algorithm with known $k \geq 1$. Assume that k is so small relative to 2^n that the chance of producing a $\left|\delta^{(j)}\right\rangle$ on each run is 1. (This is a simplifying approximation, to prevent you from getting distracted from the more important and interesting problem coming up.) You are trying to find all k solutions $\delta^{(0)}, \delta^{(1)}, \ldots, \delta^{(k-1)}$. Suppose that you've found ℓ values so far, where $0 \leq \ell \leq k-1$. Then the probability that the next run of Grover's algorithm produces a new value, that you haven't seen yet, is $(k-\ell)/k$.

1. What is the expected (average) number of runs needed to collect all k values? (You will need to use an argument as in our analysis of Simon's algorithm: expectation of the geometric distribution, combined with linearity of expectation.)

2. Show that the expected number of runs is less than or equal to $k(1+\log k)$, where log denotes the natural (base-e) logarithm. (You will need to use a little calculus!)