The final exam is *Friday, May 12 at 3:00 - 5:30 pm.* It will be closed-book, closed-notes, and will may cover material from any part of the course. The final is being held in Uris Auditorium.

We will hold a review session on Wednesday, May 10 at 8:00 pm in Phillips 101.

The problems below are meant to help in studying for the final, and we will discuss them at this review session. You can also discuss the problems with us in office hours, which will be held according to the normal schedule up through the final.

(1) Let $G = (V, E)$ be an undirected graph. We say that a set of vertices $X \subseteq V$ is a *dominating set* if every vertex of G is a member of X or is joined by an edge to a member of X. (Note the difference between a dominating set and a vertex cover.)

(a) If a set C is a vertex cover, must it also be a dominating set? Give a proof or a counter-example.

(b) If a set X is a dominating set, must it also be a vertex cover? Give a proof or a counter-example.

(c) Show that the following problem is NP-complete: Given a graph G and a natural number k, does G have a dominating set of size at most k?

(d) Give a polynomial-time algorithm that takes a graph G and produces a dominating set of size at most $O(\log n)$ times the minimum size.

(2) Suppose we are given a tree T, rooted at a given node r. Each node v has a *weight* w_v, and our goal is to find an independent set in T of maximum possible weight. Show how to solve this in polynomial time.

(3) We define the *evacuation problem* as follows. We are given a directed graph $G = (V, E)$ (picture a network of roads); a certain collection of k nodes $X \subseteq V$ are designated as *populated nodes*; and a certain other collection of $\ell > k$ nodes $C \subseteq V$ are designated as *candidate nodes.* (Assume that X and C are disjoint.)

We declare a set of k nodes $S \subseteq C$ to be *safe* if there is a set of "evacuation routes" from the populated nodes X to S — we define a set of evacuation routes as a set of paths in G so that (i) each node in X is the tail of exactly one path, (ii) each node in S is the last node in exactly one path, and (iii) the paths do not share any edges. Such a set of paths gives a way for the occupants of the populated nodes to "escape" to S, without overly congesting any edge in G.

Given G, X, and C, show how to decide in polynomial time whether there exists a safe set $S \subseteq C$.

(4) At a lecture in a computational biology conference I attended about a year ago, a well-known protein chemist talked about the idea of building a "representative set" for a large collection of protein molecules whose properties we don't understand. The idea would be to intensively study the proteins in the representative set, and thereby learn (by inference) about all the proteins in the full collection.

To be useful, the representative set must have two properties.

- It should be relatively small, so that it will not be too expensive to study it.
- Every protein in the full collection should be "similar" to some protein in the representative set. (In this way, it truly provides some information about all the proteins.)

More concretely, there is a large set P of proteins. We define similarity on proteins by a *distance function* d — given two proteins p and q, it returns a number $d(p, q) \geq 0$. In fact, the function $d(\cdot, \cdot)$ most typically used is the *edit distance*, or *sequence alignment* measure, which we looked at in Chapter 4.5. We'll assume this is the distance being used here. There is a pre-defined distance cut-off Δ that's specified as part of the input to the problem; two proteins p and q are deemed to be "similar" to one another if and only if $d(p, q) \leq \Delta$.

We say that a subset of P is a *representative set* if for every protein p, there is a protein q in the subset that is similar to it — i.e. for which $d(p, q) \leq \Delta$. Our goal is to find a representative set that is as small as possible.

(a) Give a polynomial-time algorithm that approximates the minimum representative set to within a factor of $O(\log n)$. Specifically, your algorithm should have the following property: if the minimum possible size of a representative set is s^*, your algorithm should return a representative set of size at most $O(s^* \log n)$.

(b) Note the close similarity between this problem and the Center Selection problem we covered in Chapter 2.5. Why doesn't the algorithm described there solve the current problem?