CSCI 570, Summer 2016 Homework 5 Solutions

1. In a graph G, a DOMINATING SET in a subset $V' \subseteq V$ of the vertices such that every vertex either is in V' or is adjacent to one that is. Prove that the problem of determining if a dominating set of size k is present in a general graph is $\mathcal{NP}$-complete.

   DOMINATING SET $\in \mathcal{NP}$: Given a set of vertices V', we can check, for every vertex in the original graph, if it is in V'. If it is not, we can check each of its neighbors. This total runtime is $O(n^2m)$.
   DOMINATING SET is $\mathcal{NP}$-hard: Suppose I wanted to solve VERTEX COVER and had access to a solution to DOMINATING SET. I will create a new graph G', which will be G, minus any vertices with degree zero, plus some additional vertices and edges. For each edge $e = (u, v)$ in the original graph, I can create a new vertex uv in G' with edges to u and v. I now check if there is a dominating set of size k in G' and return accordingly.
   No false positives: Any dominating set of size k in G' is a vertex cover of size k in the original graph, as it must include either u, v, or uv. If you needed to produce the vertex cover itself, and any new vertex uv is in the dominating set, "move" it to u or v. If both u and v are already in the dominating set, you have a smaller vertex cover.
   No false negatives: Similarly, any vertex cover in G of size k is a dominating set of the same size in G'. Because every edge $e = (u, v)$ has at least one endpoing in the cover, then vertex uv is adjacent to a chosen one. Furthermore, every vertex with degree greater than 0 is either included, or at least one of its neighbors is included (as there's an edge from it to each neighbor).

2. Sometimes you can know people for years and never really understand them. Take your friends Raj and Alanis, for example. Neither of them is a morning person, but now they're getting up at 6AM every day to visit local farmers' markets, gathering fresh fruits and vegetables for the new health-food restaurant they've opened.

   In the course of trying to save money on ingredients, they've come across the following problem. There's a large set of n possible raw ingredients they could buy $I_1, I_2, \ldots I_n$. Ingredient I_j must be purchased in units of size s_j grams, costs c_j dollars per unit, and is safe to use for t_j days from the date of purchase.

   Over the next k days, they want to make a set of k different daily specials, one each day. The order in which they schedule the specials is up to them. The ith daily special uses a subset $S_i \subseteq \{I_1, I_2, \ldots, I_n\}$ of the raw ingredients. Specifically, it requires $a_{i,j}$ grams of ingredient I_j. Furthermore, the ingredients are partitioned into two subsets: those that must be purchased on the very day the special is offered, and those that can be used until they expire.

   This is where the opportunity to save money on ingredients comes up. Often, when they buy a certain ingredient I_j, they don't need the whole thing for the special they're making that day. Thus, if they can follow up quickly with another special that uses I_j, but doesn't require it to be fresh that day, they can save money by not having to purchase I_j again. Of course, scheduling the basil recipes so close together makes it harder to schedule the goat cheese recipes close together, and so forth – that's where the complexity comes in.

   So we define the DAILY SPECIAL SCHEDULING problem as follows: given data on ingredients and recipes as above, and a budget x, is there a way to schedule the k daily specials so that the total money spent on ingredients over the course of all k days is at most x?

   Prove that DAILY SPECIAL SCHEDULING is $\mathcal{NP}$-complete.

DAILY SPECIAL SCHEDULING $\in \mathcal{NP}$: the certificate is the order of the specials, and the verifier ensures that the total money spent to serve in this order is at most the budget constraint.
To finish the proof that this is $\mathcal{NP}$-complete, we will show how to use a solution to it to solve HAMILTONIAN PATH. Because we have to order the specials in the problem, and the vertices in HAMILTONIAN PATH, we can reason that we should create one special for each vertex. For each edge, we will create an ingredient, to be sold two grams at once for \$1, and will have a shelf life of two days. We set our budget to be $2m - n + 1$. I claim there is a way to schedule these specials iff there was a Hamiltonian Path in the original graph.
If there is such a path, we can use that ordering for the specials. We will order each edge's ingredients once; if these were completely independent, this would cost us $\$2m$. However, we reuse $n - 1$ of them, at a savings of \$1 each time we do that. This gives us a total cost of $\$2m - (n - 1) = 2m - n + 1$
The converse also holds: a special ordering with that cost means that exactly $n-1$ ingredients were reused, so there must be a chain of one ingredient in common for each set of consecutive specials, and thus a Hamiltonian Path.