CSE 6331 Homework 7

Due: Thursday, October 26 by class time

1. [Coin Changing] Let $A_n = \{a_1, a_2, \ldots, a_n\}$ be a set of distinct coin types (e.g., $a_1 = 50$ cents, $a_2 = 25$ cents, $a_3 = 10$ cents, etc). Note that a_i may be any positive integer and $a_1 > a_2 > \cdots > a_n$. Each type is available in unlimited quantity. Given A_n and an integer $C > 0$, the coin changing problem is to make up the exact amount C using a minimum total number of coins.

   (a) Show that if $a_n \neq 1$ then there exists an A_n and C for which there is no solution to the changing problem.

   (b) Show that if $a_n = 1$ then there is always a solution.

   (c) When $a_n = 1$, a greedy method to the problem will make change by using coin types in the order $a_1, a_2, \ldots, a_n$. When coin type a_i is being considered, as many coins of this type as possible will be used. Show that this algorithm doesn't necessarily generate an optimal solution.

   (d) Prove that if $A_n = \{k^{n-1}, k^{n-2}, \ldots, k^0\}$ for some $k > 1$, then the above greedy method always yields an optimal solution.
   Hint: Let $X = (x_{n-1}, \ldots, x_1, x_0)$ be the greedy solution and let $Y = (y_{n-1}, \ldots, y_1, y_0)$ be any optimal solution such that

   $$C = \sum_{i=0}^{n-1} x_i k^i = \sum_{i=0}^{n-1} y_i k^i.$$

   Show that $x_i = y_i$ for all $0 \leq i \leq n-1$. Note: $k^m = \sum_{i=0}^{m-1} (k-1)k^i + 1$.

2. Let $G = (V, E)$ be an undirected graph. A subset $U \subseteq V$ is called a *node cover* if each edge in E is incident upon at least one node in U. Finding a minimum node cover for a general graph is NP-hard, but if the graph is a tree, then a minimum node cover can be obtained by the greedy method. Design a greedy algorithm that always generates an optimal solution. (Explain your algorithm in plain English.)

3. Consider the Activity problem discussed in class. Suppose now we want to maximize the *total sum* of selected intervals, $\sum_{i \in A} (f_i - s_i)$, where A is the set of selected intervals. Solve this problem using any method. Your algorithm must be $O(n^2)$.