CS:5350 Homework 5, Spring 2016

Due at exam time on Thu, Mar 3

Collaboration: You are welcome to form groups of size 2 and work on your homeworks in groups. Of course, you are not required to work in groups. Every group should make one submission and names of both group members should appear on the submission and both students in a group will receive the same score. Other than the TA and the professor, you can only discuss homework problems with your group partner. Collaboration can be positive because talking to someone else about these problems can help to clarify your ideas and you will also (hopefully) get to hear about different ways of thinking about the problem. On the other hand, collaboration can be negative if one member of the group rides on work being done by the other member – please avoid this situation. If your solutions are (even partly) based on material other than what has been posted on the course website, you should clearly acknowledge your outside sources.
Late submissions: No late submissions are permitted. You will receive no points for your submission if your submission is not turned in at the beginning of class on the due date.
Evaluation: Your submissions will be evaluated on correctness *and* clarity. Correctness is of course crucial, but how clearly you communicate your ideas is also quite important.

1. A *dominating set* of a graph $G = (V, E)$ is a vertex subset $D \subseteq V$ such that every vertex $v \in V$ is either in D or has a neighbor in D. A *minimum dominating set (MDS)* of a graph G is a dominating set of G with fewest number of vertices. No one knows how to solve the problem of computing an MDS on arbitrary graphs in polynomial time. (In fact, since MDS is NP-complete, it is expected that MDS cannot be solved in polynomial time.) But, for *trees* there is a simple and efficient dynamic programming algorithm for MDS. This algorithm is similar to the polynomial-time maximum independent set problem on trees that we discussed in class on Tue, 2/23.

   In this problem you don't have to describe the complete algorithm. Simply state the problems you will solve for a rooted tree (rooted at a vertex v) and then state recurrence relations that express these problems in terms of problems on subtrees rooted at children of v. Just like in the maximum independent set solution, you may have to define problems closely related to MDS.

2. (This problem is from "Algorithm Design" by Kleinberg and Tardos.) Each month, you can run your business out of an office in New York (NY) or out of an office in San Francisco (SF). In month i, you'll incur a cost of N_i if you operate out of NY and a cost S_i if you operate out of SF. However, if you run your business out of one city in month i and then out of the other city in month $i + 1$, you incur a *moving cost* of M for switching offices. Given a sequence of n months, a *plan* is a sequence of n locations – each one equal to NY or SF – such that the i^{th} location indicates the city you'll be based in in the i^{th} month. The *cost* of a plan is the sum of the operating costs for each of the n months, plus a moving cost of M each time you switch cities. The plan can begin in either city.

   Problem: Given a value for the moving cost M, and sequences of operating costs $N_1, N_2, \ldots, N_n$ and $S_1, S_2, \ldots, S_n$, find a plan of *minimum* cost. You can assume that all costs (moving and operating) are positive.

   Example: Suppose $n = 4$, $M = 10$, and the operating costs are given by the following table:

NY	1	3	20	30
SF	50	20	2	4

   Then the plan of minimum cost would be the sequence of locations (NY, NY, SF, SF), with total cost $1 + 3 + 10 + 2 + 4 = 20$. Note that the 10 appears in the above expression because we switch locations once.

(a) Suppose that we produce a plan by simply picking, for each month i, $1 \leq i \leq n$, a location (NY or SF) that has cheaper operating cost that month. (If the operating costs are the same for the two cities in a month, then the location for that month is chosen arbitrarily.) Show that this algorithm does not produce a plan of minimum cost, by constructing an input for which it returns a plan that has greater cost than an optimal plan.

(b) Give an efficient algorithm that takes values n, M, and sequences of operating costs $N_1, N_2, \ldots, N_n$ and $S_1, S_2, \ldots, S_n$ and returns the *cost* of an optimal plan.
