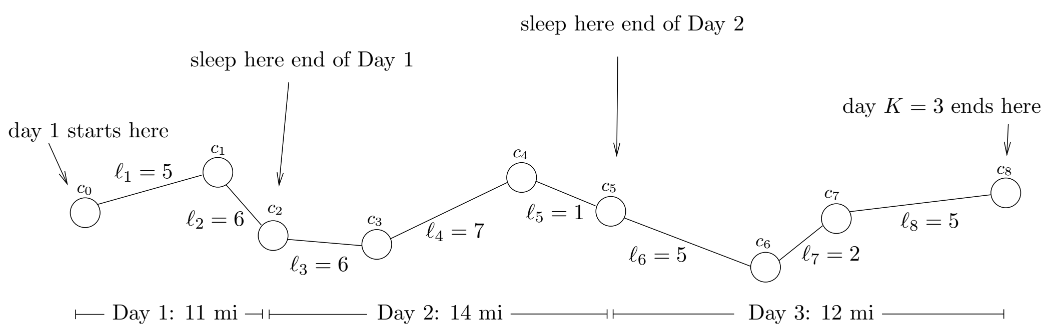
CMSC451 Homework 4
Spring 2017 due March 31, 2017 at 3pm EDT

For all algorithms, provide time complexity analysis as well as a formal proof of correctness. Homework solutions should be clearly written and electronically submitted via ELMS (`http://elms.umd.edu`) by the due date listed above.

1. Recall the Knapsack problem in which there are n items, each item i with nonnegative weight w_i and a distinct value v_i. You are interested in finding the subset S of maximum value $\sum_{i \in S} v_i$ whose weight $\sum_{i \in S} w_i$ does not exceed budget W. In class, we saw how to find the maximum value achievable by any subset. Design an $O(nW)$-time algorithm to construct the set S of maximum value.

2. We are going on a trip along the Appalachian trail. We have a list of all possible campsites $c_0, \ldots, c_n$ that we can camp at along the way. We want to do this trip in exactly K days, starting at c_0 and finishing at c_n, stopping for $K-1$ nights to camp. The distances ℓ_i between adjacent campsites c_{i-1} and c_i are specified in advance, and we can only set up camp at a campsite. How can we plan this trip to minimize d, where d is the maximum amount of walking done in a single day?

   For example, if our trip involves three days of walking, and we walk 11, 14, and 12 miles on each day respectively, then the most we have walked per day is 14 (see figure). Another itinerary that involves walking 11, 13, and 13 miles on each day has cost 13.

   Given K and the distances ℓ_i, provide an $O(n^2K)$-time algorithm that computes the minimum d, such that on each day, we walk at most d miles, reaching our destination c_n by the Kth day.

3. Given a string of characters $c_1 \ldots c_n$, we say that a substring $c_i \ldots c_j$ for $1 \leq i \leq j \leq n$ is a *palindrome* if it reads the same forward and backwards. For example, "abacaba" is a palindrome. Give an $O(n^2)$-time algorithm to find the longest palindrome substring in the input string $c_1 \ldots c_n$.

4. You are given a rectangular piece of cloth with dimensions $X \times Y$, where X and Y are positive integers, and a list of n types of products that can be made using the cloth.

For each product type i, you know the needed cloth dimensions $a_i \times b_i$ and the selling price c_i. Assume that a_i, b_i and c_i are all positive integers. You have a machine that can cut any rectangular piece of cloth into two pieces either horizontally or vertically.

Design a polynomial-time algorithm that determines the best return on the $X \times Y$ piece of cloth, that is, a strategy for cutting the cloth so that the products made from the resulting pieces give the maximum sum of selling prices. You are free to make as many copies of a given product as you wish, or none, if desired. You may find the following hints helpful:

(a) First, prove that without loss of generality cuts are made only on integer boundaries.

(b) Develop a recurrence for the maximum return that can be obtained from a cloth of arbitrary integral dimension $i \times j$. (The natural thought would be: *we either use the cloth for a single copy of a product, or ...*)

(c) Use this recurrence to design the algorithm.