EE 360C — Algorithms — Summer 2013
Homework #8

Due: August 14, 2013 11:30am (in class)

Homework problems are to be done individually. You may discuss the problem and general concepts with other students, but you must write your solutions independently.

Each question is worth 15 points. Maximum possible score is 30.

Whenever you give an algorithm, prove that it is correct.

---

1. As some of you know well, and others of you may be interested to learn, a number of languages (including Chinese and Japanese) are written without spaces between the words. Consequently, software that works with text written in these languages must address the *word segmentation problem*—inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like "meetateight" and deciding that the best segmentation is "meet at eight" (and not "me et at eight," or "meet ate ight," or any of a huge number of even less plausible alternatives). How could we automate this process?

   A simple approach that is at least reasonably effective is to find a segmentation that simply maximizes the cumulative "quality" of its individual constituent words. Thus, suppose you are given a black box that, for any string of letters $x = x_1 x_2 \dots x_k$, will return a number $quality(x)$. This number can be either positive or negative; larger numbers correspond to more plausible English words. (So $quality(\text{“}me\text{”})$ would be positive, while $quality(\text{“}ght\text{”})$ would be negative.)

   Given a long string of letters $y = y_1 y_2 \dots y_n$, a segmentation of y is a partition of its letters into contiguous blocks of letters; each block corresponds to a word in the segmentation. The total quality of a segmentation is determined by adding up the qualities of each of its blocks. (So we'd get the right answer above provided that $quality(\text{“}meet\text{”}) + quality(\text{“}at\text{”}) + quality(\text{“}eight\text{”})$ was greater than the total quality of any other segmentation of the string.)

   Give an efficient algorithm that takes a string y and computes a segmentation of maximum total quality. (You can treat a single call to the black box computing $quality(x)$ as a single computational step.)

2. Consider the following inventory problem. You are running a company that sells some large product (let's assume you sell trucks), and predictions tell you the quantity of sales to expect over the next n months. Let d_i denote the number of sales you expect in month i. We'll assume that all sales happen at the beginning of the month, and trucks that are not sold are *stored* until the beginning of the next month. You can store at most S trucks, and it costs C to store a single truck for a month. You receive shipments of trucks by placing orders for them, and there is a fixed ordering fee of K each time you place an order (regardless of the number of trucks you order). You start out with no trucks. The problem is to design an algorithm that decides how to place orders so that you satisfy all the demands $\{d_i\}$, and minimize the costs. In summary:

   - There are two parts to the cost: (1) storage—it costs C for every truck on hand that is not needed that month; (2) ordering fees—it costs K for every order placed.
   - In each month you need enough trucks to satisfy the demand d_i, but the number left over after satisfying the demand for the month should not exceed the inventory limit S.

   Give an algorithm that solves this problem in time that is polynomial in n and S.