CS314: Algorithms
Homework 3

1. You are consulting for a trucking company that does a large amount of shipping between Boston and New York. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight w_i. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive, since otherwise a customer might get upset at the unfair ordering. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

   But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is how they are thinking. Maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed.

   Prove that for a given set of boxes with specified weights w_i that the greedy algorithm currently in use actually minimizes the number of trucks that are needed. (Hint: Your proof should follow the same type of analysis as we used for the interval scheduling problem in class: it should establish optimality of this greedy packing algorithm by identifying a measure under which it "stays ahead" or all other solutions.)

2. Consider a long, quiet country road with houses scattered very sparsely along it. (Picture this road as a long line segment, with an eastern endpoint and a western endpoint.) Further, suppose that despite this bucolic setting, the residents of the houses are avid cell phone users. You want to place cell phone base stations at points along the road so that every house is within four miles of one of the base stations. Give an efficient algorithm that achieves this goal, using as few base stations as possible.

3. The wildly popular Spanish-language search engine El Goog needs to do a serious amount of computation every time it recompiles its index. Forturnately, the company has at its disposal a single large supercomputer, together with an essentially unlimited supply of high-end PCs.

   They've broken the overall computation into n distinct jobs, labeled $J_1, J_2, \ldots, J_n$, which can be performed completely independently of one another. Each job consists of two stages: first it needs to be preprocessed on the supercomputer, and then it needs to be finished on one of the PCs. Our notation will be that job J_i needs p_i (or $P[i]$, if you prefer to think of these as input arrays) seconds on the supercomputer, followed by f_i (or $F[i]$) seconds on a PC.

   Since there are at least n PCs available on the premises, the finishing of the jobs can be done fully in parallel, so that all the jobs can be processed on PCs at the same time. However, the supercomputer can only work on one job at a time, so the system managers need to work out an order in which to feed jobs to the supercomputer. As soon as the first job is done on the supercomputer, it will be handed to a PC for finishing; at that point in time, a second job can be started on the supercomputer. When the second job is finished on the supercomputer, it can proceed immediately to a PC for finishing regardless of whether the first job is done or not (since PCs work in parallel), and so on.

   A *schedule* is an ordering of the jobs for the supercomputer, and the *completion time* of the schedule is the earliest time at which all jobs are finished processing on PCs. This is an important quantity to minimize, since it determines how rapidly El Goog can generate a new index.

   Give a polynomial time algorithm (as fast as possible) that finds a schedule which is as small as possible.

4. **Extra credit:** Suppose you are a simple shopkeeper living in a country with n different types of coins, with values $1 = c[1] < c[2] < \cdots < c[n]$. (In the U.S., $n = 6$ and the values are 1,5,10,25,50, and 100 cents.) Your beloved and benevolent dictator, El Generalissimo, has decreed that whenever you give a customer change you must use the smallest possible number of coins, so as not to wear out the image of him so lovingly engraved on each coin.

   (a) In the U.S., there is a simple greedy algorithm that always results in the smallest number of coins: subtract the largest coin which is not too large and recursively give change for the reminder. El Generalissimo does not approve of American capitalist greed, however. Show that there is a set of coin values for which the greedy algorithm does not always give the smallest possible number of coins.

   (b) Now suppose El Generalissimo decides to impose a currency system where the coin denominations are consecutive powers $b^0, b^1, b^2, \ldots, b^k$ of some integer $b \geq 2$. Prove that despite El Generalissimo's disapproval, the greedy algorithm described in part (a) does make optimal change in this system as well.