I500/B609: Fundamental Computer Concepts of Informatics

Discussion problems (week 7, Greedy Algorithms)

1. Your friends are starting a security company that needs to obtain licenses for *n* different pieces of cryptographic software. Due to regulations, they can only obtain these licenses at the rate of at most one per month. Each license is currently selling for a price of $100. However, they are all becoming more expensive according to exponential growth curves: in particular, the cost of license *j* increases by a factor $r_j > 1$ each month, where r_j is a given parameter. This means that if license *j* is purchased *t* months from now, it will cost $100\, r_j^t$. We will assume that all the price growth rates are distinct; that is, $r_i \neq r_j$ for licenses $i \neq j$ (even though they start at the same price of $100). Given n rates of price growth $r_1, r_2, \ldots, r_n$ of *n* software, devise an algorithm to find an optimal order in which to buy the licenses so that the total amount of money spent is minimized.

2. Define a *unit interval* as a subset of the real line of the form [a, b] = {z : a ≤ z ≤ b}, where b−a=1. For example, [5/4, 9/4] is a unit interval. Consider the problem Interval-Cover where the input is a set {x1, x2, . . . , xn} of n points in an arbitrary order on the real line, and the output is a set consisting of the smallest number of unit intervals that cover all the input points (in the sense that every point is in at least one of the intervals). As an example, suppose the input is {7/4, 7/2, ½, 2/3, 2, 0}. Then {[0, 1], [5/4, 9/4], [5/2, 7/2]} is a valid output: we can easily check that these three intervals cover all of the six input points, and also that no two unit intervals can cover all of the input points. (a) Consider the following greedy strategy for the Interval-Cover problem. First select a unit interval that by itself covers the maximum number of input points. Then remove the input points that are covered by this interval and repeat this procedure with the remaining input points until none remain. Give an example of an input to Interval-Cover where any algorithm that follows the above procedure will fail to find an optimal selection of unit intervals. (b) Give an efficient greedy algorithm that solves the Interval-Cover problem. Your algorithm should be explained so that it is easy to understand. (c) What is the running time of your algorithm? (d) Argue that your algorithm is correct.

3. Your friend is working as a camp counselor, and he is in charge of organizing activities for a set of junior-high-school-age campers. One of his plans is the following mini-triathalon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles. The plan is to send the contestants out in a staggered fashion, via the following rule: the contestants must use the pool one at a time. In order words, first one contestant swims the 20 laps, gets out and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as the second is out and starts biking, a third

contestant begins swimming … and so on.

Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10 miles of bicycling), and a project running time it will take him or her to complete the 3 miles of running). Your friend wants to decide on a schedule for the triathalon: an order in which the contestants will be sent. Lets say the completion time of a schedule is the ealiest time at which all contestants will be finishing with all three parts of the triathalon, assuming they will each spend exactly their projected time on swimming, biking and running. What is the best order for sending people out, if one wants the whole competition to be over as early as possible? More precisely, give an algorithm that produces a schedule whole completion time is as small as possible.

4. A small business – say, photocopying service with a single large machine faces the following scheduling problem. Each morning, they get a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps their customers happiest. Customer i's job will take t_i time to complete. Given a schedule (i.e., an ordering of the jobs), let C_i be the finishing time of job i. For example, if job j is the first to be done, we would have $C_j=t_j$. Each customer i also has a given weight w_i, that represents his or her importance to the business. The happiness of customer i is expected to be dependent on the finishing time of i's job. So the company decides that they want to order the jobs to minimize the weighted sum of the completion times $\Sigma_j w_j C_j$..

   Example: Suppose there are two jobs: $t_1=1$ and $w_1= 10$, $t_2=3$ and $w_2=2$. Then doing job 1 first would yield a weighted completion time of $10\cdot 1+2\cdot 4 = 18$ while doing the second job first would yield the larger weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$. Thus, the first schedule is better.

   Design an efficient algorithm to solve this problem. That is, you are given a set of n jobs with processing time t_i and weight w_i, or each job i. You want to order the jobs so as to minimize the weighted sum of the completion time.