Pre-tutorial questions

Do you know the basic concepts of this week's lecture content? These questions are only to test yourself. They will not be explicitly discussed in the tutorial, and no solutions will be given to them.

1. Greedy algorithms
   (a) What is typical with a greedy approach?
   (b) What is the Interval Scheduling problem?
   (c) What is the Interval Partitioning problem?
   (d) What is the Scheduling to Minimize Lateness problem?
   (e) What is the Caching problem?
   (f) To prove correctness of a greedy algorithm we often use an "exchange argument". What is the idea of an exchange argument?

Tutorial

Problem 1

Suppose we are to schedule print jobs on a printer. Each job j has associated a weight $w_j > 0$ (representing how important the job is) and a processing time t_j (representing how long the job takes). A schedule σ is an ordering of the jobs that tell the printer how to process the jobs. Let C_j^σ be the completion time of job j under the schedule σ.

Design a greedy algorithm that computes a schedule σ minimizing the sum of weighted completion times, that is, minimizing $\sum_j w_j C_j^\sigma$.

Problem 2

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-triathlon exercise: each contestant must swim 20 laps of a pool, then cycle 10 km, then run 3 km. 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 other words, first contestant swims the 20 laps, gets out, and starts biking. As soon as the first contestant is out of the pool, the second contestant begins swimming the 20 laps; as soon as he/she's out and starts cycling, 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 *cycling time* (the expected time it will take him or her to complete the 10 km of cycling), and a projected *running time* (the time it will take him or her to complete the 3 km of running. Your friend wants to decide on a *schedule* for the triathlon: an order in which to sequence the starts of the contestants. Let's say that the *completion time* of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts.

P1

Decreasing ratio

P2

Decreasing run + bike time

1 Problem 2

Let the contestants be numbered $1, 2, ..., n$, and let s_i, b_i, r_i represent the respective swimming, biking and running times of contestant i. Our greedy algorithm is to send contestants out in order of decreasing $b_i + r_i$.

Let X be the schedule produced by the greedy algorithm and let Y be an optimal schedule. If $X = Y$, we are done. Otherwise, assume that $X \neq Y$, so Y must have two contestants i, j, where j is sent out directly after i, but $b_i + r_i < b_j + r_j$. Consider the schedule obtained by swapping the ith and jth contestants. In this swapped schedule, i will start biking after j has gotten out of the pool. i will finish faster in this schedule than j will finish in the previous schedule since $b_i + r_i < b_j + r_j$. We can conclude that our swapped schedule does not have greater completion time than Y.

What's the best order for sending people out, if one wants the whole competition to be over as early as possible? More precisely, give an efficient algorithm that produces a schedule whose completion time is as small as possible. Prove the correctness of your algorithm.

Problem 3

Assume that you are given n white and n black dots, lying on a line. The dots appear in any order of black and white. Design a greedy algorithm which connects each black dot with a (different) white dot, so that the total length of wires used to form such connected pairs is minimal. The length of wire used to connect two dots is equal to their distance along the line.

Problem 4

Consider a post office that sells stamps in four different denominations: 1c, 4c, 16c, 64c. Design and analyze a greedy algorithm that given a positive integer n, finds the smallest set of stamps whose total value equals n.

Problem 5

[**Advanced**] The k-centre problem in the Euclidean plane is defined as follows. Given a set V of n points in the Euclidean plane find a subset S of V of size k such that the maximum Euclidean distance of any point in V to its closest point in S is minimized. Consider the following trivial algorithm:

1. Pick any arbitrary point $p \in V$, set $S \leftarrow \{p\}$.
2. While $|S| < k$, find the point whose minimum distance to S is maximum and add it to S.

Prove that the above algorithm is a 2-approximation algorithm.