CS3510-A: Design and Analysis of Algorithms, Spring 2021

Homework-6

February 23, 2021

DUE DATE: Tuesday, March 2, 11:59pm

Note-1: Your homework solutions should be electronically formatted as a single PDF document that you will upload on Gradescope. If you have to include some handwritten parts, please make sure that they are very clearly written and that you include them as high resolution images.
Note-2: Please think twice before you copy a solution from another student or resource (book, web site, etc). It is not worth the risk and embarrassment.
Note-3: You need to **explain/justify** your answers. Do not expect full credit if you just state the correct answer.
Note-4: You will get 2 extra points if you submit electronically typed solutions instead of hand-written.

Problem-1 (30 points)

You are consulting for Cyclotron, a company that does a large amount of shipping between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed maximum amount of weight they are allowed to carry, C. Boxes arrive at the New York facility one by one, and each package i has a weight p_i. The facility is quite small, so at most one truck can be at the station at any time. Cyclotron requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after theirs make it to Boston faster. The company uses a simple greedy algorithm for packing: Boxes are packed in the order they arrive, and whenever the next box does not fit they send the truck on its way.

The board of directors is wondering if they might be using too many trucks, and they have hired you to see whether the situation can be improved. Their thinking is as follows: "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."

You must prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use minimizes the number of trucks needed to ship these boxes. Your proof should establish the optimality of this greedy packing algorithm by identifying a measure under which it "stays ahead" of all other possible solutions.

Solution

Problem-2 (40 points)

Your friend is working as a camp counselor, and they are in charge of organizing activities for a set of campers. One of their plans is the following mini-triathlon exercise: each camper must swim 20 laps of a pool, then bike 10 miles, then run 3 miles. The plan is to send the campers out in a staggered fashion, via the following rule: the campers must use the pool one at a time. In other words, first one camper swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second camper begins swimming the 20 laps; as soon as they are out and starts biking, a third camper begins swimming...and so on.

Each camper has a projected swimming time (the expected time it will take them to complete the 20 laps), a projected biking time (the expected time it will take them to complete the 10 miles of bicycling), and a projected running time (the time it will take them to complete the 3 miles of running). Your friend wants to decide on an order in which to sequence the starts of the campers. Let's say that the completion time of an order is the earliest time at which all campers will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on each of these three parts. Give an algorithm that produces an order whose completion time is as small as possible in $O(nlog(n))$ time.

Note: You must show your algorithm's running time with respect to the number of campers.
Note: You must show your algorithm's optimality.

Solution

Problem-3 (30 points)

Consider the following variation on the Interval Scheduling Problem. You have a supercomputer that can operate 24 hours a day, every day. People submit requests to run **daily** jobs on the supercomputer. Each such job request includes a fixed and pre-determined *start time* and an *end time*. If a job request is accepted it must run continuously, every day, for the period between the requested start and end times. Jobs can begin before midnight and end after midnight. (This makes for a type of situation different from what we saw in the Interval Scheduling Problem.) The supercomputer can run at most one job at any given point in time; no two jobs can run concurrently.

Given a list of n such job requests, provide an algorithm to accept as many job requests as possible (regardless of their length). Your algorithm should have a running time that is $O(n^2)$. You may assume for simplicity that no two jobs have the same start or end times.

Example. Consider the following four jobs, specified by (start-time, end-time) pairs.

$$(6P.M., 6A.M.), (9P.M., 4A.M.), (3A.M., 2P.M.), (1P.M., 7P.M.).$$

The optimal solution would be to pick the two jobs (9P.M.,4A.M.) and (1P.M., 7P.M.), which can be scheduled without overlapping.

Note: You must show your algorithm's running time is $O(n^2)$.

Solution