Algorithms: Test 3 (Take-home test)

Problem #1.

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 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 other 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 he or she 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 projected running time (the time it will take him or her to complete the 3 miles 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. (Again, note that participants can bike and run simultaneously, but at most one person can be in the pool at any time.)

How would you determine the best order for sending n persons out if you want the whole competition to be over as early as possible?

(i) Design an efficient O(n log n) greedy algorithm that can produce a schedule for n persons with a minimal final completion time. Write down the algorithm and prove that the algorithm always correctly provides a minimal-complete-time schedule in O(n log n) time.
(ii) Design an alternative algorithm you can think of. Write down the algorithm and prove that the algorithm always correctly provides a minimal-complete-time schedule too. What is the time complexity of your alternative algorithm?

(iii) Consider the following problem instance. What would the minimal-complete-time schedule generated by your algorithms?

	Person A	Person B	Person C	Person D
Swimming time	20	35	40	25
Biking time	25	20	20	30
Running time	20	15	30	25

See Problem #2 on the next page.

Problem #2.

Given n jobs $J_1, J_2, ..., J_n$ where each job J_i takes T_i units of time to complete and has a priority weight W_i, you need to schedule the jobs to be done in some order one by one by assigning a starting time S_i to each job J_i. Time starts from the time point 0 at the time line. If job J_i starts at time S_i , it will be completed at the time $S_i + T_i$. No two jobs can overlap in time and you want to minimize the weighted sum of the completion time $\Sigma_{n>= i >=1} W_i * (S_i + T_i)$

(i) Design an efficient O(n log n) greedy algorithm that can produce a schedule for the n jobs and minimize the weighted sum of completion time. Write down the algorithm and prove that the algorithm can always correctly produce a minimal weighted-sum-of-completion-time schedule.

(ii) Design an alternative algorithm you can think of. Write down the algorithm and prove that the algorithm can always correctly produce a minimal weighted-sum-of-completion-time schedule too. What is the time complexity of your alternative algorithm?

(iii) Consider the following problem instance. What would be the minimal weighted-sum-of-completion-time schedule generated by your algorithms?

	Job 1	Job 2	Job 3	Job 4	Job 5
W_i	4	5	2	3	6
T_i	6	7	5	6	8