CSE 6140 - Practice Session 1

1 Why is the pool always so busy anyway?

Georgia Tech is trying to raise money for the Technology Square Research Building and has decided to host the "Tech Swim Run Bike" (TSRB) Triathalon to start off the fundraising campaign!

Usually in a triathalon all athletes will perform the three events in order (swimming, running, then biking) asynchronously, but unfortunately due to some last minute planning, the race committee was only able to secure the use of one lane of the olympic pool in the campus recreation center. This means that there will be a bottleneck during the first portion (the swimming leg) of the race where only one person can swim at a time. The race committee needs to decide on an ordering of athletes where the first athlete in the order will swim first, then as soon as the first athlete completes the swimming portion the next athlete will start swimming, etc. The race committee, not wanting to wait around for an extremely long time for everyone to finish, wants to come up with a schedule that will minimize the time it takes for everyone to finish the race. Luckily, they have prior knowledge about how fast the athletes will complete each portion of the race, and they have you, an algorithm-ista, to help!

Specifically, for each athlete, i, they have an estimate of how long the athlete will take to complete the swimming portion, s_i, running portion, r_i and biking portion, b_i. A schedule of athletes can be represented as a list of athletes (e.g. $[athlete_7, athlete_4, ...]$) that indicate in which order the athletes will start the race. Using this information, they want you to find an ordering for the athletes to start the race that will minimize the time taken for everyone to finish the race, assuming all athletes perform at their estimated time. More precisely, give an efficient algorithm that produces a schedule of athletes whose completion time is as small as possible and prove that it gives the optimal solution using an **exchange argument**.

Keep in mind that once an athlete finishes swimming, they can proceed with the running and biking portions of the race, even if other athletes are already running and biking. Also note that all athletes have to swim first (i.e. some athletes won't start off running or biking). For example: if we have a race with 3 athletes scheduled to go in the order $[2, 1, 3]$, and the finishing time for athlete i is represented as a_i, then the finishing times would be as follows: (with a total finishing time of $\max(a_1, a_2, a_3)$)

$$a_2 = (s_2) + r_2 + b_2$$
$$a_1 = (s_2 + s_1) + r_1 + b_1$$
$$a_3 = (s_2 + s_1 + s_3) + r_3 + b_3$$

Review session notes

Problem statement

We first restate the problem in a reduced format so that we can reason about it more easily:

- Each athlete has a run time r_i, bike time b_i, and swim time s_i.
- Each athlete must swim, then bike, then run in order to complete the race.
- Only one athlete may swim at a time. Any number can bike and run at the same time.
- We must create an ordering of the athletes (really an ordering of swimmers) that minimizes the time that the last athlete finishes.
- We must prove that our algorithm creates an optimal ordering through an **exchange argument**.

Note: this exchange argument is covered in detail from section 4.3 in Kleinberg and Tardos (and this question is adapted from one of the practice questions there).

Key observations for deriving the greedy algorithm

- The last athlete is going to start his bike+run part after every other person has finished swimming.
  - What happens if we put the slowest athlete (in terms of bike+run time) last? (they will be the last to finish)
  - What happens if we put the fastest athlete last? (they *might* not finish last)
- **We should order the athletes by their bike time + run time, then schedule them from slowest to fastest.**

Steps for exchange argument proof

1. Assume that there is an optimal scheduling, O
   - The general strategy is to modify O, preserving its optimality until it is the schedule found by our algorithm.
2. Define how you modify O, this is the "exchange" part of the argument
   - First we let $t_i = r_i + b_i$ for each athlete i.
   - We say there is an inversion when there are 2 *consecutive* athletes (say j and k) scheduled where j is ahead of k, but $t_k < t_j$ (i.e. the first athlete is the faster of the two)
   - By construction, our greedy algorithm will not have any inversions.
3. Prove that all schedules without inversions have the same max-lateness
   - Super duper important.

- Assume for the sake of contradiction they don't, i.e. that there exists two schedules without an inversion that do *not* have the same max lateness.
- These two schedules could only differ where there are several athletes with the same bike+run times.
- But the ordering of these athletes does not affect the max lateness - a contradiction.

4. Prove that there exists an optimal schedule without inversions
    - This is where the "exchange" comes into play.
    - In the case that our "optimal schedule" has inversions, we want to show that we can swap the positions of the two athletes and decrease the max lateness. (if our optimal schedule doesn't have inversions we don't have to do anything)
    - Let i and j be the two inverted athletes, i.e. $t_i < t_j$.
        - Let f_i be the finish time for i, $f_i = s_i + t_i$.
        - Let f_j be the finish time for j, $f_j = s_i + s_j + t_j$.
        - Note: we can ignore all the swimming time before i and j, it will be a constant that cancels in next step (let $C = \sum_{k=1}^{i-1} s_k$, you can convince yourself that if you include C in both finish times they will cancel.)
    - If we swap the position of these athletes, the new finish times are:
        - $f_i' = s_i + s_j + t_i$
        - $f_j' = s_j + t_j$
    - We want to show that the max lateness is better after we swap. Here f_j was the worst time (max lateness) before the swap, so we want to show both 1.) $f_i' < f_j$ and 2.) $f_j' < f_j$:
        1. $f_i' = (s_i + s_j) + t_i$, $f_j = (s_i + s_j) + t_j$, $t_i < t_j$, therefore $f_i' < f_j$.
        2. $f_j' = (s_j) + t_j$, $f_j = s_i + (s_j) + t_j$, $t_j < s_i + t_j$, therefore $f_j' < f_j$.
5. In step 2 we defined an inversion as between two consecutive athletes. In order to get to a solution without any inversions we will swap at most $O(n^2)$ pairs of consecutive athletes. If we perform this process, we have shown that our optimal solution will have as many inversions as our greedy solution (zero), and will not have become worse in the process. Gg.