CS41 Homework 7

This homework is due at 11:59PM on Monday, October 25. This is a 7 point homework. Write your solution using LaTeX. Submit this homework using **github** as a **.tex** file. This is a **partnered homework**. You should primarily be discussing problems with your homework partner.

It's ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab teammate *while in lab.* In this case, note (in your **homework submission poll**) who you've worked with and what parts were solved during lab.

1. **Summer camp triathlon.** (K& T 4.6)
   Your friend is working as a camp counselor, and is in charge of organizing activities for a set of junior-high-school-age campers. One of the 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 person is out of the pool, a second contestant begins swimming the 20 laps; as soon as the second person 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 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 expected time it will take them to complete the 3 miles of running). Your friend wants to decide on a *schedule* for the triathalon: 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 triathalon, 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.) 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.

2. **Find the missing integer** (CLRS 4-2)
   Suppose $n = 2^k - 1$ for some k.

   An array $A[1 \dots n]$ contains all the integers from 0 to n except one. Each integer from 0 to n is represented as a k-bit string. It would be easy to determine the missing integer in $O(n)$ time by using an auxiliary array $B[0 \dots n]$ to record which numbers appear in A. Unfortunately, we cannot access an entire integer in A with a single operation. Because the elements of A are represented in binary, the only operation we can use to access them is "fetch the j^{th} bit of $A[i]$", which takes constant time. This means that reading every digit of every number in A would take $O(nk) = O(n \log n)$ operations.

   In this problem, we'll develop an efficient divide and conquer algorithm that identifies the missing integer, using only $O(n)$ operations.

   (a) If one number x is missing, it must be the case that either $x < n/2$ or $x \geq n/2$. Describe how to figure out which of these is true using only $O(n)$ operations.

(b) After you figure out whether $x < n/2$ or $x \geq n/2$, which bit(s) of x do you know?

(c) Define the sets:

$$A_{\text{small}} = \{y \in A \mid y < n/2\}$$
$$A_{\text{big}} = \{y \in A \mid y \geq n/2\}$$

We'd like to use the insight from part (2a) to intelligently decide which elements to put in A_{big} and which to put in A_{small}. This will be our preprocessing step to set up the "divide" part of our divide and conquer algorithm. Describe a way to keep track of which entries of A belong to either A_{small} and A_{big}, using only $O(n)$ work.

(d) Put together the two parts above into an algorithm that recurses on either A_{small} or A_{big}. Part (2c) should help you determine your "divide" step, and part (2b) should help you determine how to "combine" the recursive return value with new information to figure out x.

Describe your algorithm with low-level pseudocode.

(e) Write a recurrence for the runtime of this algorithm and solve it using partial substitution.