CS 3000: Algorithms & Data — Summer 1 '20 — Tim LaRock

Homework 1
Due Due Monday May 11 at 11:59pm Boston time via Canvas

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.

- This assignment is due Due Monday May 11 at 11:59pm Boston time via Canvas. Make sure to submit something before the deadline.

- Solutions must be typeset in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.

- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.

- Finding solutions to homework problems on the web, or by asking students not enrolled in the class, is strictly forbidden.

Problem 1. *Inductive Proofs*

(a) Prove the following statement by induction: For every $n \in \mathbb{N}$, $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Solution:

(b) Prove the following statement by induction: For every $n \in \mathbb{N}$, $\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$

Solution:

(c) In class I showed a plot that implies the following statement is true: "polynomials are smaller than exponentials." Specifically, $n^a = O(b^n)$ for every $a > 0$ and $b > 1$. In this problem you will use induction to prove a special case of this fact, that $n^2 = O(2^n)$, by induction.

Prove by induction that, for every $n \geq 4$, $n^2 \leq 2^n$.

Solution:

Problem 2. *Asymptotic Order of Growth*

(a) Rank the following functions in increasing order of asymptotic growth rate. That is, find an ordering $f_1, f_2, \ldots, f_{10}$ of the functions so that $f_i = O(f_{i+1})$. No justification is required.

$$n^{5/2} \qquad 4^{\log_2 n} \qquad n! \qquad 7^n \qquad \log_2(n!)$$
$$2^{3n} \qquad n^2 \log_2(n) \qquad 8n \qquad 3^{\log_5 n} \qquad \log_2(n^3)$$

Solution:

(b) Consider the following piece of code.

Algorithm 1: Waste some time
Function $A(n)$:   Let m be the smallest power of 2 that is at least n $(m = 2^{\lceil \log_2 n \rceil})$ **For** $i = 1, \ldots, m^3$: Do an operation

Give an asymptotic expression for the number of operations done by $A(n)$ as a function of n in $\Theta(\cdot)$ notation. Justify your answer. Your expression should be as simple as possible—for example, $\Theta(n)$ would be a better than $\Theta(100n + 10)$.

Solution:

Problem 3. *What Does This Code Do?*

You encounter the following mysterious piece of code.

Algorithm 2: Mystery function

Function $C(a, n)$**:**
    If $n = 1$ **:**
        | **Return** $(1, a)$
    ElseIf $n = 2$ **:**
        | **Return** $(a, a \cdot a)$
    ElseIf *n is odd* **:**
        | $(u, v) \leftarrow C(a, \lfloor \frac{n+1}{2} \rfloor)$
        | **Return** $(u \cdot u, u \cdot v)$
    ElseIf *n is even* **:**
        | $(u, v) \leftarrow C(a, \lfloor \frac{n+1}{2} \rfloor)$
        | **Return** $(u \cdot v, v \cdot v)$

(a) What are the results of $C(a, 3)$, $C(a, 4)$, and $C(a, 5)$. You do not need to justify your answers.

    Solution:

(b) What does the code do in general? Prove your assertion by induction on n.

    Solution:

(c) In this problem you will analyze the running time of C as a function of n. Prove that, for every $n \in \mathbb{N}$, the number of multiplication operations performed in evaluating $C(a, n)$ is at most $2 \cdot \log_2(n - 1) + 1$ (where we use the convention that $\log_2(0) = 0$).

    Solution:

Problem 4. *IRL Divide and Conquer*

A professor teaching an in-person class wants to count the number of students physically present in the class. They came up with two plausible ways to count. In this question, you will analyze the correctness and running time of these two solutions with respect to n, the number of students in the class.

Algorithm 3: SimpleCounting

Find first student
First student says 1
While *Students remain*
   | Find the next student
   | Next student says (what last student said + 1)

(a) A *loop invariant* is a condition that holds true before and after every iteration of a loop. We can often use loop invariants to help explain why an algorithm must produce correct output. Identify such a loop invariant in SimpleCounting and use it to explain (informally) why the algorithm is correct.

   Solution:

(b) What is the running time of SimpleCounting with respect to the number of students in the class n? Provide your answer in big-O notation and explain in words why this is the case.

   Solution:

Algorithm 4: FancyCounting

Everyone initialize their number to 1
Everyone stand up
While *Someone is standing*
   | Try to pair with your neighbor
   | **If** *You are not in a pair* :
   | | Stay standing
   | **Else**
   | | Sum up your numbers
   | | Sit down if you are the tallest person in the pair
If you are standing, say your number

(a) As above, identify an appropriate loop invariant in FancyCounting and use it to explain why the algorithm is correct. It would be sufficient, but is **not** necessary, to provide a formal proof of correctness.

   Solution:

(b) Write down a recurrence relation that describes the asymptotic runtime of FancyCounting as the number of students in the class (n) grows.

   Solution:

(c) Based on your recurrence relation above, what is the asymptotic runtime of FancyCounting? Optional: Prove it by induction for a bonus point.

Solution:

(d) Optional: For a bonus point, explain what it is about Zoom that makes it very difficult to run FancyCounting. Hint: The answer is **not** because people can't stand up; replace "stand up" with "raise hand."

Solution: