COSC 240, Fall 2018: Problem Set #3

Assigned: Monday, 10/1
Due: Wed, 10/10, at the beginning of class (hand in hard copy).
Lectures Covered: 8 to 9 (with some leftover problems from 7).
Academic Integrity: You can work with other people in the class but you must write up your own answers in your own words. You can also use the textbook and talk to the professor. You may not use any other resources (e.g., material found online) or talk to people outside the class about these problems. See the syllabus for details on the academic integrity policy for problem sets.

Divide-and-Conquer Problems

1. Consider the problem in which you are provided an array of bits and the goal is to count the number of 0 bits in the array. Describe with pseudocode a randomized divide-and-conquer algorithm that satisfies the following two properties:

   (a) For the worst-case random choices, it requires $\Omega(n^2)$ steps.

   (b) Its randomized step complexity is $\Theta(n)$.

   Your algorithm must be fully recursive and cannot use any loops. You are allowed, however, to call a subroutine **busy-work**(A, i, j), that when passed the input array A of size n, and two indices $i \leq j \leq n$, has a step complexity of $\Theta(j - i + 1)$, and does not modify any values in A.

   (Hint: My solution flipped a biased coin at the beginning of the initial call—i.e., when the subproblem size is the same as the size of the whole array—and then passed the outcome of this one flip to all recursive calls that followed. The problem is also solvable without passing an initial coin flip outcome to subsequent recursive calls, but the analysis for these styles of solutions is slightly more complicated.)

2. Formally analyze the randomized step complexity of your answer to the above problem.

Amortized Analysis Problems

Assume you run a testing center where students arrive throughout the day to take the GRE. (For simplicity, assume students never leave). Your testing center has a classroom of size 2^i, for each $i \geq 0$. You only have one proctor, however, so all students at the test center at any given time must be in the same classroom. You have also noticed that students perform poorly if a room seems empty, so you insist on the rule that the room containing the students at any given time must have more full than empty seats.

To accommodate these constraints you use the following algorithm to handle each new arrival of a student:

- Start the day using the smallest available classroom (which is size $2^0 = 1$).
- Once you run out of space in a classroom of size 2^i (e.g., the room is full and then a new student arrives), move all of the existing students and the new student to the classroom of size $2 \cdot 2^i = 2^{i+1}$.

You are concerned about the cost of this room scheduling policy. In particular, you calculate the cost of each student arrival to be the number of students that must be moved (including the new student) to accommodate the arrival. More formally, consider a sequence of n arrivals. Let c_i be the cost of the i^{th} student arrival. You can define $c_i = 1 + m_i$ where:

$$m_i = \begin{cases} i-1 & \text{if } i-1 \text{ is a power of 2} \\ 0 & \text{otherwise} \end{cases}$$

The three questions that follow ask about this above scenario. The purpose of these questions is to explore how the three types of amortized analysis studied in class might help you more accurately understand the arrival costs over time in this scenario.

1. Prove that arrivals have a constant amortized cost by using the *aggregate analysis* technique discussed in class (and Chapter 17.1 of the textbook).

2. Prove that arrivals have a $\log n$ amortized cost, assuming n total students arrive, by using the *accounting method* technique discussed in class (and Chapter 17.2 of the textbook).

3. Consider a simple unary counter that counts from 1 to k before wrapping back around to 1 (for some $k \geq 1$). One way to implement this counter is with an array A of size k. Initially $A[1] = 1$ and $A[j] = 0$, for $2 \leq j \leq k$. A variable p keeps track of the smallest position in the array that currently stores a 0. If there are no 0's in the array, then $p = k + 1$. We initialize $p \leftarrow 2$.

   To INCREMENT the counter, there are two cases. If $p = k + 1$, loop through A and set each position to 0, then set $A[1] \leftarrow 1$ and $p \leftarrow 2$. Otherwise, if $p \leq k$, then set $A[p] \leftarrow 1$ and $p \leftarrow p + 1$.

   Let c_i be the cost of the i^{th} INCREMENT. If we focus only on changing array entries when calculating cost, we can use the following definition for c_i:

$$c_i = \begin{cases} 1 & \text{if } i \text{ is not a multiple of } k, \\ \text{k+1} & \text{else.} \end{cases}$$

   In the worst case, therefore, $c_i = k + 1$. Prove that INCREMENT has constant amortized cost using the potential method.

4. **Optional Extra Credit.** Earlier, you were asked to use the accounting method to prove a logarithmic amortized cost for arrivals. Refine your answer to use the accounting method to now prove a constant amortized cost.

   (Hint: students who arrive after a classroom expansion might want to help the students who arrived before the expansion regain the credit they will need to pay for their movement in the next expansion.)