COSC 240, Fall 2017: Problem Set $\#3$

Assigned: Thursday, 9/28/17.
Due: Wed, 10/11, at the beginning of class (hand in hard copy).
Lectures Covered: 8 to 10.
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.

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 four questions that follow ask about this above scenario. The purpose of these questions is to explore how amortized analysis might help you more accurately understand the arrival costs.

1. To better understand the worst case behavior of this scheduling algorithm answer the following:
   (a) Provide a concise and clear argument why the definition of c_i properly captures the arrival cost of new students.
   (b) Argue that in a sequence of n operations, $c_i = \Theta(n)$ in the worst case.
2. Prove that arrivals have a constant amortized cost by using the *aggregate analysis* technique discussed in class (and Chapter 17.1 of the textbook).
3. 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).

4. Refine your answer from the preceding problem to show that arrivals have constant amortized cost using the accounting method.

   (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.)

5. In class, we discussed the aggregate analysis and accounting method techniques for amortized analysis. A third such technique is called the *potential method*. To prepare for this question, please read the discussion of this method in Chapter 17.3 of your textbook. Then answer the following question:

   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.

6. In the lecture notes on hashing, I introduced the *load factor* α as a way to measure the potential performance of a hash table with m entries containing n total elements. Assume I claim to have a great hash function h that guarantees the following: if you use h to implement a hash table with chaining, then every list is always strictly smaller than the load factor. Prove this is impossible.

7. At the end of the lecture notes on hashing, I discussed implementing hash tables with open addressing. Open addressing requires an extended hash function that maps each key to a permutation of the values $0, 1, ..., m - 1$. Consider the following simple definition of an extended hash function h:

$$\forall k \in U : h(k) = \langle 0, 1, 2, ..., m - 1 \rangle.$$

   Given an open address hash table implemented with h, how many probes are required for an unsuccessful SEARCH, assuming that $\alpha < 1$?