CSCI 3104 Spring 2023
Instructors: Ryan Layer and Chandra Kanth Nagesh

Homework 27

Due Date ..... April 20, 2023
Name ..... **Blake Raphael**
Student ID **109752312**
Collaborators ..... **Alex Barry, Brody Cyphers, and Ben Kohav**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Gradescope page** only (linked from Canvas). Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (I agree to the above, Blake Raphael). □

3 Standard 27 - Doubling and Amortized Analysis

Problem 2. A dynamic array-list is a data structure that can support an arbitrary number of append (add to the end) operations by allocating additional memory when the array becomes full. The standard process covered in class is to double (adds n more space) the size of the array each time it becomes full. You cannot assume that this additional space is available in the same block of memory as the original array, so the dynamic array must be copied into a new array of larger size. Here we consider what happens when we modify this process. The operations that the dynamic array supports are

- `Indexing A[i]`: returns the `i`-th element in the array
- `Append(A,x)`: appends `x` to the end of the array. If the array had `n` elements in it (and we are using 0-based indexing), then after `Append(A, x)`, we have that `A[n]` is `x`.

27(a)

a. Consider a dynamic array-list that adds 7 to its length (that is, increases length from n to $n+7$) each time it's full. Derive the amortized runtime of Append for this data structure. Clearly explain the reasoning behind your calculations.

Answer. Consider the following hash table A:
We begin with A at size 1.
After our first insertion, we resize A to A' at $n+7$ or 8.
We continue insertions until we hit 8 and have to resize $A' \to A''$ at $n+7$ or 15.
This means we resize the table $\frac{n-1}{7}$ times.

Our number of operations is the following:

$$\begin{aligned}\sum_{i=1}^{n} C_i &\leq n + \sum_{j=1}^{\frac{n-1}{7}} 7j \\ &\leq n + \frac{1}{14}(n-1)(n+6) \\ &= \Theta(n^2).\end{aligned}$$

Since $\Theta(n^2)$ is the runtime of a list with n items, we divide by n to get the amortized runtime of append. So we get $\Theta(n)$ as the amortized runtime of Append.

□

27(b)

b. Derive the amortized runtime of Append for a dynamic array that squares its length (that is, increases length from n to n^2) each time it's full. Derive the amortized runtime of Append for this data structure. Clearly explain the reasoning behind your calculations.

Answer. Consider the following hash table A:
We begin with A at size 2.
After our second insertion, we resize A to A' at n^2 or 4.
We continue insertions until we hit4 and have to resize $A' \to A''$ at n^2 or 16.
This means we resize the table $\log \log n$ times.

Our number of operations is the following:

$$\begin{aligned}
\sum_{i=1}^{n} C_i &\leq n + \sum_{j=1}^{\log \log n} n^{\frac{1}{2^k}} \\
&\leq n + \sum_{j=1}^{\log \log n} n^{\frac{1}{2}} \\
&\leq n + n^{\frac{1}{2}} \log \log n \\
&= \Theta(n).
\end{aligned}$$

Since $\Theta(n)$ is the runtime of a list with n items, we divide by n to get the amortized runtime of append.
So we get $\Theta(1)$ as the amortized runtime of Append.

□