CSCI 3104 Spring 2023
Instructors: Prof. Layer and Chandra Kanth Nagesh

Midterm 2 Standard 15 - Analyzing Code III: Writing down recurrences

Due Date ..... April 28th
Name ..... **Blake Raphael**
Student ID **109752312**
Quiz Code (enter in Canvas to get access to the LaTeX template) ..... **PuGzk**

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 Canvas page** only. 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 **may not collaborate with other students**. **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, Discord, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.

2 Standard 15 - Analyzing Code III: Writing down recurrences

2.1 Problem 1

Problem 1. Write down a recurrence for the runtime complexity of this algorithm. Clearly justify your answer. You are **not** being asked to solve the recurrence.

```
foo(n) {
    if (n <= 2) {
        return 0
    }

    foo(n/4)
    foo(n/4)
    foo(n/4)
    foo(n/4)
    for (j = 1; j < n; j*4) {
        print()
    }
}
```

Answer.

$$T_n = \begin{cases} \Theta(1) & : n \leq 2, \\ 4T(n/4) + \Theta(n) & : n > 1. \end{cases}$$

We start with our base case which is if $n \leq 2$, we just return. This gives us a runtime of $\Theta(1)$. Next we dig into the recursive part of the code. We are recursively calling foo 4 times and dividing n by 4 each time we do so, then we add the runtime of the non-recursive code on the end. This gives us $4T(n/4) + \Theta(n)$.

□