Ashcon Abae

CMSC 451

3 November 2019

Homework 2

1. Considering $f(n) = 3n^2 + 5$ and $g(n) = 53n + 9$. To prove or disprove $f \in \Omega(g)$ using L'Hopital's rule, we can write:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{3n^2+5}{53n+9} = \lim_{n\to\infty} \frac{3n+\frac{5}{n}}{53+\frac{9}{n}} = \infty$$

   This implies that f(n) has a larger order of growth than g(n). Thus, $f \in \Omega(g)$.

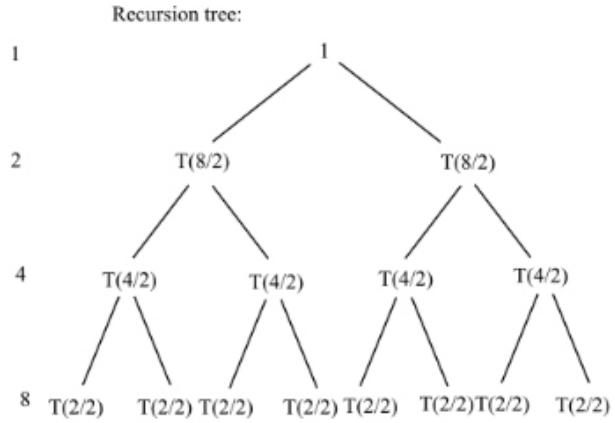
   Next, to prove or disprove $g \in \Theta(f)$ using L'Hopital's rule, we can write:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{53n+9}{3n^2+5} = \lim_{n\to\infty} \frac{53+\frac{9}{n}}{3n+\frac{5}{n}} = 0$$

   This implies that g(n) has a smaller order of growth than f(n). Thus, $g \notin \Theta(f)$

2. The following functions are listed from lowest asymptotic order to highest, with functions that are of the same order on the same line:
   a. $\log_3 n$
   b. $\log_2 n$
   c. $\sqrt{n}$
   d. $10n + 7$
   e. $n \log_2 n$
   f. $n^2 + 5n + 10$
   g. $n^3 + 2n + 1,\ \ n^3 + 5n$
   h. 2^n
   i. 3^n

3. The recursion tree for when n = 8 for the described recursive function sum() is as follows:



a. A formula for determining the number of nodes would be 2n – 1, where n is the size of our array. For example, for when n = 8, our recursion tree will have 2(8) – 1 = 15 nodes
b. Because the sum function is a recursive function that calls itself twice and has "mid" calculated n constant time each recursion, we can write our execution time as follows:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1) = \Theta(n)$$

c. Because our recursion tree is actually a binary tree, the height of our tree can be calculated as $H = \log_2 n$
d. To determine our space complexity, let's review: whenever a recursive function is called, the variables are pushed onto the stack and remain in the stack until the function is completed. Thus our space complexity is proportional to the maximum depth of our recursion tree. In each call, sum function takes constant space $\Theta(1)$ for storing pointer of array, first, last, mid, and return address. Additionally, we can evaluate our maximum depth as $\Theta(\log n)$ times. Thus, the maximum space complexity is as follows:

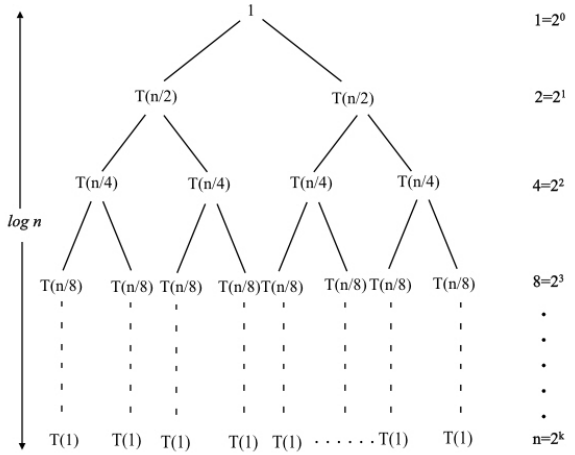
$$\Theta(1) * \Theta(\log n) = \Theta(\log n)$$

e. The following is an iterative solution for this same problem:

```
int sum(int[] array, int first, int last) {
    int total = 0;
    for(int i = 0; i <= last; i++;) {
        total = total + array[i];
    }
    return total;
}
```

In this iterative version, our time complexity would be equivalent to the recursive version and would be $\Theta(n)$. That being said, the iterative version would differ from the recursive version in that its space complexity would be lower than the recursive version's space complexity.

4. Below, you can find the modified recursion tree from problem 3 to show the amount of work on each activation and the row sums. The maximum depth is displayed on the left hand side:



The sum function is a recursive function and it calls itself twice in each call. Additionally, the

function calculates mid per call. Therefore, we can write our recurrence equation as follows:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$$

The initial condition that begins our recurrence equation is the condition of if the "first" and "last" parameters of the sum function are equal. If they are not, we go through recursion. Using the Master Theorem, we have $T(n) = aT\left(\frac{n}{b}\right) + f(n)$. If we compare this to our recurrence equation above, we can see that $a = 2$, $b = 2$, and $f(n) = \Theta(1)$. Next, we calculate our critical exponent as $\log_b a = \log_2 2 = 1$. To solve this equation, we will apply case 2 of the Master Theorem:

$$f(n) = \Theta\left(n^{\log_2 2 - 1}\right) = \Theta(n^0) = \Theta(1), \text{then } T(n) = \Theta\left(n^{\log_2 2}\right) = \Theta(n)$$

This recursive solution is optimal in its time complexity. Its space complexity isn't at its most optimal; as we compared an iterative solution to the same problem in part e of problem 3 above, we could see that another solution is more optimal when comparing space complexities.