Frederick Wittman
Dr. Kotthoff
COSC 3020
10/11/19

Assignment 1

1.

Definition

"f(n) = O(g(n)) means there are positive constants c and k such that $0 \leq f(n) \leq cg(n)$ for all $n \geq k$."

Source: https://xlinux.nist.gov/dads/HTML/bigOnotation.html

Proof

The key idea here is what it means for $O(\log_{10}n)$ to be the same as $O(\log_2 n)$. This means that the two functions have the same big-O time complexity *as a function of n*. We can show that this is true by proving that $\log_2 n \leq c_1 * \log_{10}n$, and similarly that $\log_{10}n \leq c_2 * \log_2 n$, where c_1 and c_2 are positive constants that are not functions of n. Notice that this is the same as proving both that $\log_{10}n$ is $O(\log_2 n)$ and that $\log_2 n$ is $O(\log_{10}n)$. First, I show that $\log_2 n$ is $O(\log_{10}n)$. Let the constant c_1 be $\log_2 10$:

$c_1 * \log_{10}n$

$= \log_2 10 * \log_{10}n$

$= \log_2(10^{\log_{10}n})$ { property of logarithms }

$= \log_2 n$ { property of logarithms }

$\geq \log_2 n$

Second, I show that $\log_{10}n$ is $O(\log_2 n)$. Let the constant c_2 be $\log_{10}2$:

$c_2 * \log_2 n$

$= \log_{10}2 * \log_2 n$

$= \log_{10}(2 \text{ ^ } \log_2 n)$ { property of logarithms }

$= \log_{10} n$ { property of logarithms }

$\geq \log_{10} n$

Note that these two inequalities are true for each value of n for which log is defined. For this reason, we can select k in the range (0, positive infinity). Since we have shown that $\log_{10} n$ is $O(\log_2 n)$ and that $\log_2 n$ is $O(\log_{10} n)$, we conclude that $O(\log_2 n)$ is the same as $O(\log_{10} n)$.

2.

There are two key things to notice about the mystery function: the relationship between n and the number of calls to mystery(n / 3) where (n / 3) > 1, and the value of the square of the argument at each of these calls.

Where n is in the range (negative infinity, 3], the function returns a null value. For n in this range, then, the function has constant time complexity. Notice that for each value of n for which the expression ceil($\log_3 n$) increments (by 1), the calls to mystery(n/3) which were previously (n / 3) ≤ 1 (the base case calls) are now calls for which (n / 3) > 1. The number of recursive calls to mystery(n / 3) where (n / 3) ≤ 1 can therefore be modeled by $2^0 + 2^1 + \ldots + 2^{\text{ceil}(\log 3n) - 1} = 2^{\text{ceil}(\log 3n)} - 1$. A sample of this progression is illustrated below:

n	(3, 9]	(9, 27]	(27, 81]	(81, 243]
# of calls where (n / 3) > 1	$2^2 - 1 = 3$	$2^3 - 1 = 7$	$2^4 - 1 = 15$	$2^5 - 1 = 31$

For each of the calls where (n / 3) > 1, there are (n / 3)^2 operations in the for loop. Where m = ceil($\log_3 n$) - 1, the total number of operations as a function of n can therefore be modeled by:

$$n^2 + 2 * (n/3)^2 + 4 * (n/9)^2 + \ldots + 2^m * (n/3m)^2$$

$$= \square(i = 0, m)\, 2^i * (n^2 / 3^i)$$

$$= n^2 * \square(i = 0, m)\, (2^i / 3^i)$$

$= O(n^2 * log\ n)$ { m is a function of n; there are m + 1 terms in the sum }

3.

1, 7, 1, 5, 3, -1, 9
1, 1, 7, 5, 3, -1, 9
1, 1, 5, 7, 3, -1, 9
1, 1, 3, 5, 7, -1, 9
-1, 1, 1, 3, 5, 7, 9
-1, 1, 1, 3, 5, 7, 9

4.

We examine the time complexity as a function of n, as n approaches infinity, and discounting constant factors. Take first the msort function. The outer for loop executes ceil($\log_2$n) times. The for loop system in aggregate executes 1 + (n / 2 + … + n / 2^(ceil($\log_2$n) - 1) times. This reflects the fact that we pass the array as a whole once and also partition the array by twos, fours, etc and as necessary, and pass each of these sections to the merge function. Where m = 2^(ceil($\log_2$n) - 1), the total number of operations is therefore given by:

$1 + (n / 2 + \ldots + n / 2^m)$

$= 1 + \square\ (i = 1, m)\ n / 2 + \ldots + n / 2^m$

$= 1 + n * \square\ (i = 1, m)\ 1 / 2 + \ldots + 1 / 2^m$

$= 1 + n * \Theta(m)$ { the number of terms in the sum is a function of m }

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

This function is $\Theta(n * log\ n)$ in the worst, average, and best cases, because it executes all the operations detailed above without regard to the degree to which the original argument is sorted.

Now we examine the merge function. If merge is passed an already sorted list, it is Θ (1). However, in the average and worst cases a number of elements proportional to n

must be moved for each of a number of elements proportional to n. This yields $\Theta(n^2)$ complexity for the average and worst cases.

In the worst case, the merge sort algorithm I implemented is therefore:
$\Theta(n * log\ n) * \Theta(n^2) = \Theta(n^3 * log\ n)$

5.

If the first element of the list is chosen as the pivot, and the list is randomly ordered, this is the same as choosing a pivot randomly. If n is the size of the list, and as n goes to infinity, there is a 1 / n chance of the randomly chosen pivot being the median element of the list; a 2 / n chance that it will be adjacent to the median; a 2 / n chance that it will be one element away; etc. Where P(x) is the probability that the pivot will be x distance from the median of the list, this probability distribution is represented by the sequence { P(0) = 1 / n, P(1) = 2 / n, … , P(n / 2) = 2 / n }. On average, we will select a pivot that is (0 + 1 + 2 + … + (n / 2 - 2) + (n / 2 - 1) + (n / 2)) / (n / 2) = (n / 4) distance from the median. (Found by collapsing the sum.)

We now examine the median of three approach. Since the list is randomly ordered, the median of three approach suggested by the problem is equivalent to finding the median of three random elements. By the central limit theorem, if we plot the median values derived from successive median of three calculations, with replacement, we will, in the limit, arrive at a normal distribution centered on the true median of the list. Since elements that are close to the true median are overrepresented in this distribution relative to the distribution from randomly selecting a pivot, which is flat, the average distance of the chosen pivot in this approach will be less than (n / 4). We can find what the average distance will be in this approach by solving for x in:

$\int (0,\ x)\ RHS\ normal\ distribution\ f(x) = \int (x,\ n / 2)\ RHS\ normal\ distribution\ f(x)$

Since a good pivot is synonymous with a pivot closer to the median of the list, we conclude that the median of three approach is more likely to pick a good pivot than simply choosing the first element of the randomly ordered list.