CS 161 Fall 2017: Section 1

Asymptotic Analysis

For each of the following functions, prove whether $f = O(g)$, $f = \Omega(g)$, or both ($f = \Theta(g)$). (For example, by specifying some explicit constants $n_0, c > 0$ (or n_0, c_1, c_2 in the case that $f = \Theta(g)$) such that the definition of Big-Oh, Big-Omega, or Big-Theta is satisfied.)

(a)	$f(n) = n \log\left(n^3\right)$	$g(n) = n \log n$
(b)	$f(n) = 2^{2n}$	$g(n) = 3^n$
(c)	$f(n) = \sum_{i=1}^{n} \log i$	$g(n) = n \log n$

Recurrence Relations

Recall the Master theorem from lecture:

Theorem 0.1. *Given a recurrence $T(n) = aT(\frac{n}{b}) + O(n^d)$ with $a \geq 1$, and $b > 1$, and $T(1) = \Theta(1)$, then*

$$T(n) = \begin{cases} O(n^d \log n) & \textit{if } a = b^d \\ O(n^d) & \textit{if } a < b^d \\ O(n^{\log_b a}) & \textit{if } a > b^d \end{cases}$$

What is the Big-Oh runtime for algorithms with the following recurrence relations?

(a) $T(n) = 3T(\frac{n}{2}) + \Theta(n^2)$

(b) $T(n) = 4T(\frac{n}{2}) + \Theta(n)$

(c) $T(n) = 2T(\sqrt{n}) + O(\log n)$

Divide and Conquer: Majority Element

Suppose we are given an array, A, of length n, with the promise that there exists some number, x, that occurs at least $n/2 + 1$ times in the array. Additionally, we are only allowed to check whether two elements are equal (no comparisons).

(a) Complete the following pseudo-code for a divide-and-conquer algorithm that returns the majority element of A. Feel free to assume that the n is a power of 2.

```
MajorityElement(Input: array A of length n)
If n = 1, return A[1]
Else
   Let m1 = MajorityElement(A[1:n/2])
   Let m2 = MajorityElement(A[n/2+1:n])
```

(b) Give a brief but formal proof of the correctness of your algorithm. Again, feel free to assume $n = 2^s$ for some integer s. [Hint: induction on s!!]

(c) Express the runtime of your algorithm via a recurrence relation, and solve the relation to give the asymptotic (Big-Oh) runtime of your algorithm.