CS 344 HW 1

Fall 2019

1. For each of the following, prove using the definition of $O(\cdot)$:
   (a) $7n + \log(n) = O(n)$
   (b) $n^2 + 4n + 7 = O(n^2)$
   (c) $n! = O(n^n)$
   (d) $2^n = O(2^{2n})$

2. For each of the following, state whether $f(n) = O(g(n))$, $g(n) = O(f(n))$, or $f(n) = \Theta(g(n))$, and prove:
   (a) $f(n) = n^2$ $\quad$ $g(n) = n^3$
   (b) $f(n) = \log_2 n$ $\quad$ $g(n) = \log_3 n$
   (c) $f(n) = 2^n$ $\quad$ $g(n) = 3^n$
   (d) $f(n) = 2^n$ $\quad$ $g(n) = 2^{n+1}$

3. (CLRS 2.3-3) Use induction to prove that when n is a power of 2, then the solution of the recurrence

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

   is $T(n) = n \lg n$.

4. (CLRS 2.3-4) (Note: here arrays are 1-indexed) We can express insertion sort as a recursive procedure as follows. To sort $A[1..n]$, we recursively sort $A[1..n-1]$ and then insert $A[n]$ into the sorted array $A[1..n-1]$.
   (a) Write a recurrence for the worst-case running time of this recursive version of insertion sort.
   (b) Use the recursion tree method to solve the recurrence and find a $O(\cdot)$ bound on the worst-case running time.

5. Suppose we modify merge sort to split the array into 4 non-overlapping subarrays at each stage.
   (a) What is the running time of merging 4 arrays, in $O(\cdot)$ notation?
   (b) Write the resulting recurrence relation for merge sort and solve for the running time. Does it improve on the traditional algorithm?