CS 4349.006 Homework 2

Due Thursday, September 16 on eLearning

Please solve the following **2** problems.

1. Using Θ-notation, provide asymptotically tight bounds in terms of n for the solution to each of the following recurrences. Assume each recurrence $T(n)$ has a non-trivial base case of $T(n) = \Theta(1)$ for all $n < n_0$ where n_0 is a suitably large constant. For example, if asked to solve $T(n) = 2T(n/2) + n$, then your answer should be $\Theta(n \log n)$. **Give a brief explanation for each solution.**

   (a) $A(n) = 4A(n/2) + n^2$

   (b) $B(n) = 7B(n/2) + n^3$

   (c) $C(n) = 5C(n/3) + n$

   (d) $D(n) = D(n/4) + D(n/2) + n$

   (e) $E(n) = 3E(n/3) + n \lg n$

   Advice: One of the three common cases discussed in Lecture 5 still applies for part D, even though the subproblem sizes are not equal. For part E, solve it without the $\lg n$ *factor at the end and then see what the* $\lg n$ *factor does to each of the recursion tree level sums.*

2. An ***inversion*** in an array $A[1 .. n]$ is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. The number of inversions in an n-element array is between 0 (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted backward). Describe and analyze an algorithm to count the number of inversions in an n-element array in an $O(n \log n)$ time. **Your analysis on this and future algorithm design problems need only argue for a good big-Oh bound unless we say otherwise. Don't forget to justify why your algorithm correctly counts the inversions!**

   Advice: Modify mergesort and its merge procedure so they not only sort the array but also return the number of inversions it contained before sorting. Be sure to update your count correctly as you discover inversions during the merge procedure.