Sort the following recurrences in increasing order of growth of the corresponding functions. Justify (very) briefly.¹

(a) \( T(n) = 16T(n/4) + n; \)
(b) \( T(n) = 2T(n/4) + n; \)
(c) \( T(n) = 3T(n/5) + \log n; \)
(d) \( T(n) = 9T(n/3) + n^2; \)
(e) \( T(n) = T(n/3) + 10; \)
(f) \( T(n) = 9T(n/3) + n^3; \)
(g) \( T(n) = 8T(n/2) + n^3. \)

Solution: INSERT YOUR SOLUTION HERE

¹For this entire homework assignment, you may ignore the fact that the argument to \( T \) may not be an integer.
Consider the recurrence $T(n) = T(n/3) + T(n/2) + n$.

(a) (4 Points) Using a recursion tree, determine a tight asymptotic upper bound on $T(n)$.

Solution: INSERT YOUR SOLUTION HERE □

(b) (4 Points) Prove your upper bound using induction.

Solution: INSERT YOUR SOLUTION HERE □

(c) (4 Points) Using the substitution method, solve the recurrence $U(n) = 4U(\lceil \sqrt{n} \rceil) + 1$ with $U(2) = 1$.

Solution: INSERT YOUR SOLUTION HERE □
Previous: Fall 16.

For instructive purposes, consider the recurrence

\[ T(n) = \sqrt{n}T(\sqrt{n}) + n, n > 1, T(1) = 1. \]

In an attempt to solve it, one simplifies the equation for \( T(n) \) by dividing it by \( n \):

\[ \frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 1, \]

or, equivalently, using the substitution \( S(n) = T(n)/n \),

\[ S(n) := S(\sqrt{n}) + 1 \]

Now

\[ S(n) = S(\sqrt{n}) + 1 = S(n^{1/4}) + 2 = \ldots = S(1) + (\log \log n - 1) = \log \log n, \]

and \( T(n) = n \log \log n. \)

The following three recurrences can be solved similarly by a clever change of variables:

(a) (5 Points) \( T(n) = \frac{n \cdot T(n-1)}{n+2}, n > 1, T(1) = 1; \)

**Solution:** INSERT YOUR SOLUTION HERE

(b) (5 Points) \( nT(n) = (n - 2)T(n - 1) + 3, n > 2, T(1) = T(2) = 1; \)

**Solution:** INSERT YOUR SOLUTION HERE

(c) (5 Points) \( T(n) = 3T(n - 1) - 2T(n - 2), n > 2, T(2) = 1, T(1) = 0. \)

**Solution:** INSERT YOUR SOLUTION HERE
Let $A[1 \ldots n]$ be an array of pairwise different numbers, where for simplicity you may assume that $n$ is a power of two. We call pair of indices $1 \leq i < j \leq n$ an inversion of $A$ if $A[i] > A[j]$. The goal of this problem is to develop a divide-and-conquer based algorithm running in time $\Theta(n \log n)$ for computing the number of inversions in $A$.

(a) (8 points) Suppose you are given a pair of sorted integer arrays $A$ and $B$ of length $n/2$ each. Let $C$ an $n$-element array consisting of the concatenation of $A$ followed by $B$. Give an algorithm (in pseudocode) for counting the number of inversions in $C$ and analyze its runtime. Make sure you also argue (in English) why your algorithm is correct.

Solution: INSERT YOUR SOLUTION HERE

(b) (8 points) Give an algorithm (in pseudocode) for counting the number of inversions in an $n$ element array $A$ that runs in time $\Theta(n \log n)$. Make sure you formally prove that your algorithm’s running time (e.g., write the recurrence and solve it.)

(Hint: Combine merge sort with part (a).)

Solution: INSERT YOUR SOLUTION HERE