1. **(Writing Recurrences)** Suppose you’re writing a recurrence relation to recursively describe the runtime of a Divide-and-Conquer algorithm. This recursive algorithm takes an input of size \( n \) and recurses on 9 subproblems, each of size \( \frac{n}{3} \), until it hits a base case of size 1. The work done at a top level with input size \( n \) is \( O(n^2) \) (i.e. work done to create subproblems and "merge" their solutions, but **NOT** to solve the subproblems themselves).

**Write the recurrence relation (recursive definition) that best describes the runtime of such an algorithm. You do not need to provide the base case.**

[HINT: your answer should be in a Master-Theorem-compatible form of \( T(n) = aT\left(\frac{n}{b}\right) + O(n^d) \), for some appropriate constants \( a, b, d \) that you should choose.]

2. **(Master Theorem)** Recall the Master Theorem from Lecture 3 and the recurrence relation form that suits the Master Theorem. Consider the following recurrence relation:

\[
T(n) = 5T\left(\frac{n}{3}\right) + n \quad \text{for } n \geq 3 \\
T(n) = 1 \quad \text{for } n < 3
\]

Which of the following best describes the corresponding recursion tree for this recurrence relation?

(A) “Top-heavy” (work at highest level dominates, aka \( a < b^d \))
(B) “Bottom-heavy” (work at lowest level dominates, aka \( a > b^d \))
(C) Neither (amount of work is roughly same across levels, aka \( a = b^d \))

3. **(Solving Recurrences)** What is the Big-O solution (i.e. the best/tightest answer you can give of the form \( T(n) = O(\ldots) \)) to the following recurrence relations?

**Hint:** Try using the Master Theorem first, but if the recurrence relation doesn’t have the right form, try using the Substitution Method unraveling step to form your guess.

**Note:** When we ask you to derive a Big-O solution, we expect the tightest bound solution you can provide (unless specified otherwise).

3.1. \[
T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n} \quad \text{for } n \geq 4 \\
T(n) = 1 \quad \text{for } n < 4
\]

3.2. \[
T(n) = 7T\left(\frac{n}{3}\right) + n^2 \quad \text{for } n \geq 3 \\
T(n) = 1 \quad \text{for } n < 3
\]

3.3. \[
T(n) = T(n-2) + n \quad \text{for } n \geq 2 \\
T(n) = 1 \quad \text{for } n < 2
\]

4. **(Choosing a Pivot for SELECT)** Recall from Lecture 4 that we learned about the (recommended) MEDIAN_OF_(SUB)MEDIAN approach to determine a pivot for our SELECT algorithm.
4.1. Why didn’t we instead choose the minimum or the maximum of the array as our pivot each time? [We are expecting: a very brief justification (a couple sentences should suffice).]

4.2. Why didn’t we instead choose the exact median of the array as our pivot each time? [We are expecting: a very brief justification (a couple sentences should suffice).]

4.3. What is the Big-O runtime of the PARTITION subroutine used by the linear-time SELECT algorithm presented in Lecture 4? (After a pivot was chosen, this subroutine was used to partition the remaining elements around the pivot).