Romeo Capozziello HW 4

Problem 1:

Consider the following recurrent algorithm complexity. Note that the Master Algorithm cannot be applied directly as it is not in the T(n) = aT($\frac{n}{b}$) + f(n) format.

T(n) = 2T($\frac{n}{8}$) + 2T($\frac{n}{3}$) + n

(a) Use the Master Algorithm to find the Θ complexity of the lower bound T_L(n) = 4T_L($\frac{n}{8}$) + n

Answer

(b) Use the Master Algorithm to find the Θ complexity of the upper bound T_U(n) = 4T_U($\frac{n}{3}$) + n

Answer

(c) Do the upper and lower bound Θ complexities agree? If f(n) = n^2, would your lower and upper bound Θ complexities agree?

Answer

(d) Using only the results from (a) and (b), fins the tightest complexity values (Use Big O, little o, Big Ω, or little ω) based on each result (a) and (b).

Answer

Problem 2:

Use a recurrence tree to find the Θ complexity of T(n) = 2T($\frac{n}{8}$) + 2T($\frac{n}{3}$) + n
[Hint: Look for the geometric series, as we did in class lecture and class notes]

Answer

Problem 3:

Towers of Hanoi is an Algorithm to solve the famous problem of moving disks from one peg onto another. The complexity is goven as T(n) = 2T(n-1) + 1.

(a) Explain why the Master Algorithm cannot be applied to solve its complexity.

Answer

(b) Draw a Recurrence Tree for Towers of Hannoi, to find its complexity.

Answer

Problem 4:

In this problem, we have a recurrence. $Algorithm_1$ calls $Algorithm_2$ and $Algorithm_2$ calls $Algorithm_1$, and so forth until the problem is solved. Use a recurrence Tree to find the complexity T_1(n) of $Algorithm_1$.

T_1(n) = 2T_2($\frac{n}{2}$) + n $\qquad$ T_2(n) = 2T_1($\frac{n}{2}$) + n^2

Answer