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

f(n) = n
$n^{log_b a} = n^{log_8 4} = n^{0.666...}$
Θ(n)

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

f(n) = n
$n^{log_b a} = n^{log_3 4} = n^{1.26}$
$\Theta(n^{1.26})$

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

No, the upper and lower bound complexities do not agree
If f(n) = n^2, then the upper and lower bound complexities would agree, they would be $\Theta(n^2)$

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

O($n^{1.26}$)

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]

$$
\begin{array}{cccccccccccccccccc}
 & & & & & & & & n & & & & & & & & & n \\
 & / & & & & / & & & & \backslash & & & & \backslash & & & & \\
 & \frac{n}{8} & & & & \frac{n}{8} & & & & \frac{n}{3} & & & & \frac{n}{3} & & & & \frac{11n}{12} \\
/ & / & \backslash & \backslash & / & / & \backslash & \backslash & / & / & \backslash & \backslash & / & / & \backslash & \backslash & & \\
\frac{n}{64} & \frac{n}{64} & \frac{n}{24} & \frac{n}{24} & \frac{n}{64} & \frac{n}{64} & \frac{n}{24} & \frac{n}{24} & \frac{n}{24} & \frac{n}{24} & \frac{n}{9} & \frac{n}{9} & \frac{n}{24} & \frac{n}{24} & \frac{n}{9} & \frac{n}{9} & & \frac{11^2 n}{12^2}
\end{array}
$$

r = $\frac{11}{12}$ > 1, Θ(n)

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.

Because it is not in the form of T(n) = aT($\frac{n}{b}$) + f(n),
where a is the num of splits and b how each branch is growing

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

Answer

<u>Problem 4:</u>

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$_1$.

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

$$
\begin{array}{ccccccccc}
 & & & \text{n} & & & & & \text{n} \\
 & / & & & & \backslash & & & \\
 & \frac{n}{2} & & & & \frac{n}{2} & & & \text{n} \\
/ & & \backslash & & / & & \backslash & & \\
\frac{n^2}{2^2} & & \frac{n^2}{2^2} & & \frac{n}{2^2} & & \frac{n}{2^2} & & \text{n}
\end{array}
$$

$T_1(\text{n}) = \Theta(\text{n})$