Lecture 7-8

Recurrence Relations

Recurrence

- A function defined in terms of
  - one or more base cases, and
  - itself, with smaller arguments.

$$T(n) = \begin{cases} 1 & \text{if } n = 1\ , \\ T(n-1) + 1 & \text{if } n > 1\ . \end{cases}$$

Solution: $T(n) = n$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1\ , \\ 2T(n/2) + n & \text{if } n > 1\ . \end{cases}$$

Solution: $T(n) = n \lg n + n$.

Contd...

$$T(n) = \begin{cases} 1 & \text{if } n = 2\,, \\ T(\sqrt{n}) + 1 & \text{if } n > 2\,. \end{cases}$$

Solution: $T(n) = \lg\lg n$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1\,, \\ T(n/3) + T(2n/3) + n & \text{if } n > 1\,. \end{cases}$$

Solution: $T(n) = \Theta(n \lg n)$.

Solving Recurrences

- Substitution method,
- Recursion-tree method, and
- Master method.

Substitution Method

Substitution Method

- Guess the solution.
- Use induction to find the constants and show that the solution works.
- Can be used to establish either upper or lower bounds on a recurrence.

Example: Get an upper-bound

- $T(n) = 2T(\lfloor n/2 \rfloor) + n$
- Guess: $T(n) = O(n\log_2 n)$
- Prove: $T(n) \le cn\log_2 n$
  - for an appropriate constant $c > 0$ and $n \ge m$.
- Assume: $T(n) \le cn\log_2 n$ is true for $\lfloor n/2 \rfloor$

$$
\begin{aligned}
T(n) &\le 2(\, c\lfloor n/2 \rfloor \log_2\lfloor n/2 \rfloor\,) + n \\
&\le cn\log_2(n/2) + n \\
&= cn\log_2 n - cn\log_2 2 + n \\
&= cn\log_2 n - cn + n \\
&\le cn\log_2 n \qquad\qquad \text{for } c \ge 1.
\end{aligned}
$$

Contd…

- Check for boundary condition, $n \geq m$ (= 1)
- Let $T(1) = 1$ be a boundary condition.
- Then for $n = 1$,

$$T(n) \leq cn \log_2 n$$

$$T(1) \leq 1 \log_2 1 = 0.$$

- Thus, the base case of the inductive proof fails.

Contd…

- Check for boundary condition, $n \geq m$ (= 2)
- Using $T(1) = 1$, derive recurrence for $T(2)$.
- Then for $n = 2$,

$$T(n) \leq cn \log_2 n$$

$$T(2) = 2T(\lfloor 2/2 \rfloor) + 2 = 2\,T(1) + 2 = 4$$

$$T(2) \leq c2 \log_2 2 = 2c.$$

- $c \geq 2$ satisfies the base case of $n \geq 2$.

Contd…

- Check for boundary condition, $n \geq m$ (= 3)
- Using $T(1) = 1$, derive recurrence for $T(3)$.
- Then for $n = 3$,

$$T(n) \leq cn \log_2 n$$

$$T(3) = 2T(\lfloor 3/2 \rfloor) + 3 = 2\,T(1) + 3 = 5$$

$$T(3) \leq c3 \log_2 3.$$

- $c \geq 2$ satisfies here as well.
- Finally, $T(n) \leq c\, n \log_2 n$ for any $c \geq 2$ and $n \geq 2$.

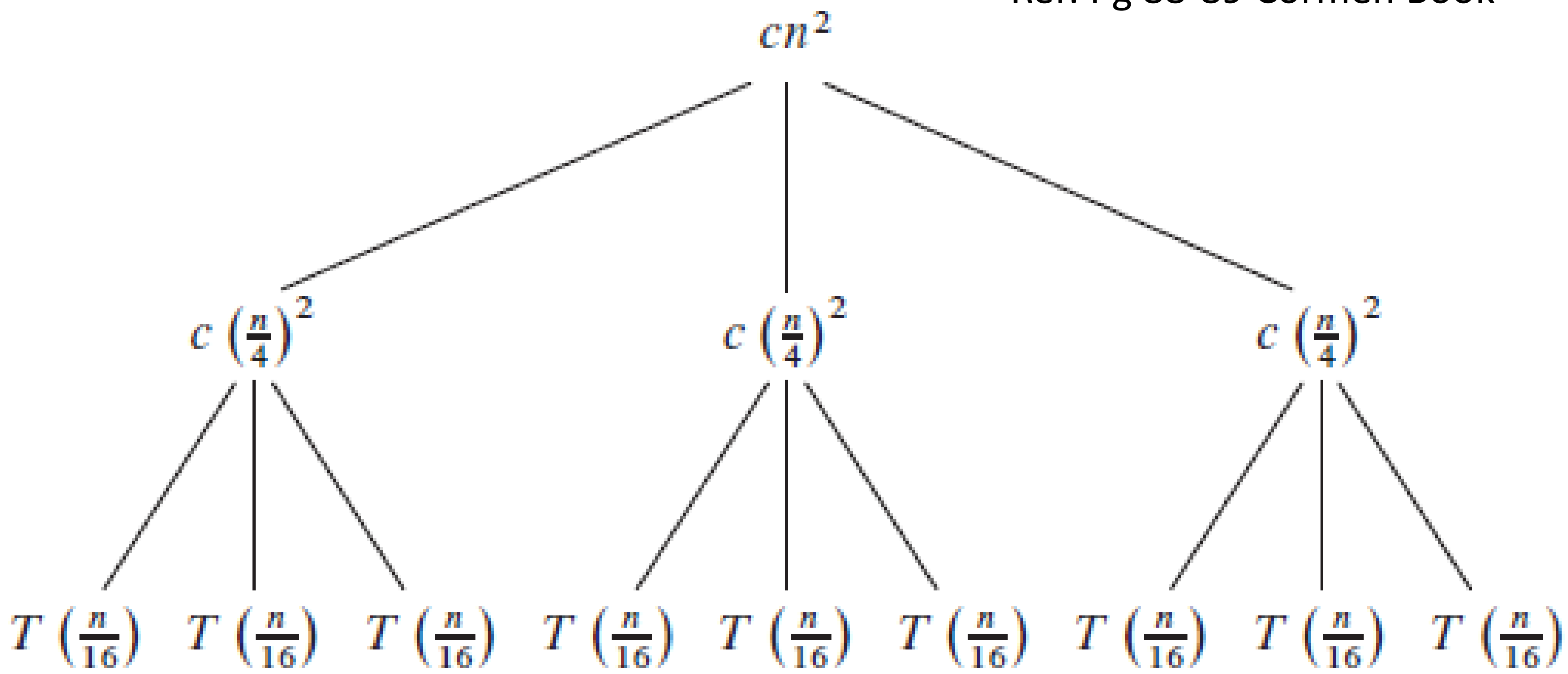
Recursion Tree Method

Recursion Tree Method

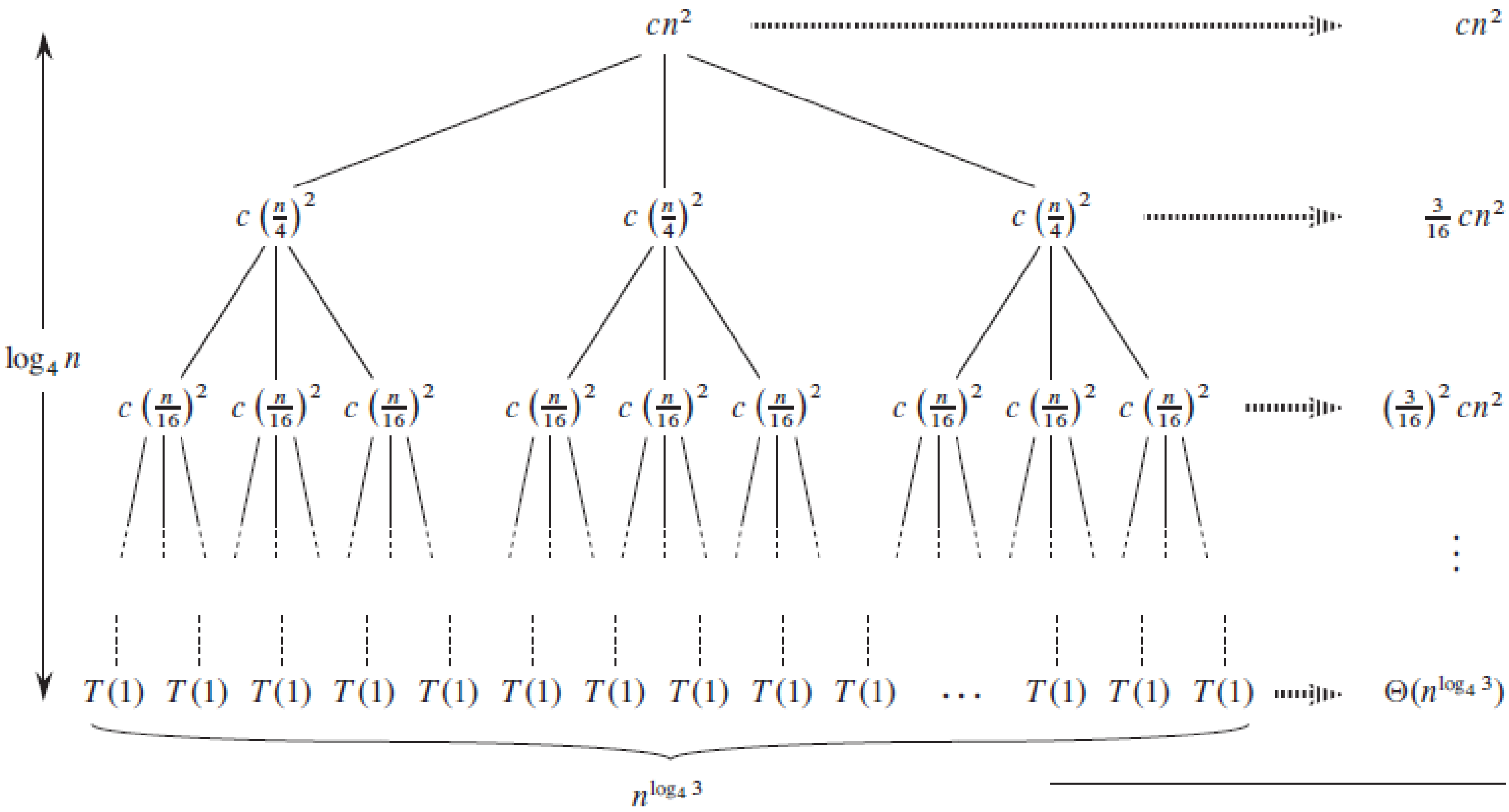
- Draw a tree describing the processing.
- Sum the amount of processing done at each level of the tree.
- Sum all of the per-level costs.

Example: $T(n) = 3T(n/4) + cn^2$

Ref: Pg 88-89 Cormen Book



Contd...


cn^2
cn^2
$c\left(\frac{n}{4}\right)^2$
$c\left(\frac{n}{4}\right)^2$
$c\left(\frac{n}{4}\right)^2$
$\frac{3}{16}cn^2$
$\log_4 n$
$c\left(\frac{n}{16}\right)^2$
$\left(\frac{3}{16}\right)^2 cn^2$
$T(1)$
$\Theta(n^{\log_4 3})$
$n^{\log_4 3}$

Contd...

- The sub-problem size decreases by a factor of 4.
- Let n be some power of 4.
- It hits n = 1 at some i, thus $n/4^i = 1 \longrightarrow i = \log_4 n$.
- Thus tree has $\log_4 n + 1$ levels (root is at i = 0).
- Cost at each depth i (= 0, 1, … $\log_4 n - 1$)
  - Number of nodes is 3^i and cost at each node is $c(n/4^i)^2$.
  - Total cost is $3^i\, c(n/4^i)^2 = (3/16)^i\, cn^2$.
- At the bottom-most level cost at each node is T(1), which is a constant. Number of nodes at the last level is $3^i = 3^{\log_4 n} = n^{\log_4 3}$. So total cost is $T(1)n^{\log_4 3} = \Theta(n^{\log_4 3})$

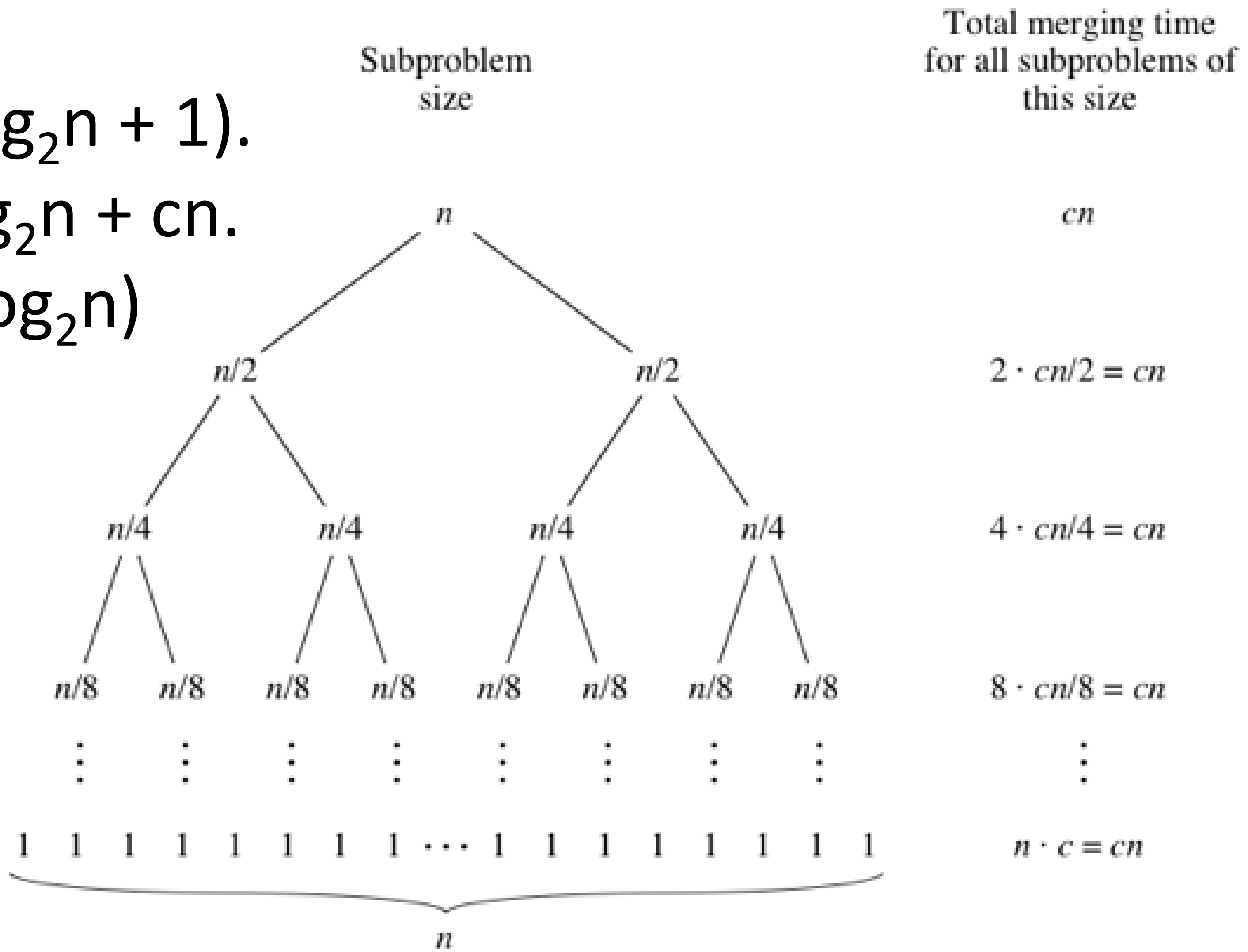
Contd...

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3}) \\
T(n) &= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)}cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13}cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

Ref: Pg 90 Cormen Book

Example: $T(n) = 2T(n/2) + \theta(n)$

$$\begin{aligned} T(n) &= cn(\log_2 n + 1). \\ &= cn\log_2 n + cn. \\ &\le O(n\log_2 n) \end{aligned}$$



Ref Pg 38
Cormen book

Master Method

Master Method

- Provides bounds for recurrences of the form

$$T(n) = aT(n/b) + f(n)$$

- where $a \geq 1$, $b > 1$, and $f(n)$ is a given function.
- Such recurrences characterizes a divide-and-conquer algorithm that
  - Creates 'a' sub-problems, each of which is '1/b' the size of the original problem.
  - Takes 'f(n)' time in the divide and combine steps together.

Master Theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ∎

Compare $n^{\log_b a}$ vs. $f(n)$

Example:

$T(n) = 4T(n/2) + n$

Reading from the equation, $a = 4$, $b = 2$, and $f(n) = n$.

Is $n = O(n^{\log_2 4-\epsilon}) = O(n^{2-\epsilon})$?

Yes, so case 1 applies and $T(n) = \theta\,(n^2)$.

Contd…

$$T(n) = 4T(n/2) + n^2$$

Reading from the equation, $a = 4$, $b = 2$, and $f(n) = n^2$.

Is $n^2 = O(n^{\log_2 4-\epsilon}) = O(n^{2-\epsilon})$?

No, if $\epsilon > 0$, but it is true if $\epsilon = 0$, so case 2 applies and $T(n) = \Theta(n^2 \log n)$.

Contd…

$T(n) = 4T(n/2) + n^3$

Reading from the equation, $a = 4$, $b = 2$, and $f(n) = n^3$.

Is $n^3 = \Omega(n^{\log_2 4+\epsilon}) = \Omega(n^{2+\epsilon})$?

Yes, for $0 < \epsilon$, so case 3 *might* apply.

Is $4(n/2)^3 \leq c \cdot n^3$?

Yes, for $c \geq 1/2$, so there exists a $c < 1$ to satisfy the regularity condition, so case 3 applies and $T(n) = \Theta(n^3)$.

Questions

1. $T(n) = 9T(n/3) + n$

For this recurrence, we have $a = 9$, $b = 3$, $f(n) = n$, and thus we have that $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$. Since $f(n) = O(n^{\log_3 9 - \epsilon})$, where $\epsilon = 1$, we can apply case 1 of the master theorem and conclude that the solution is $T(n) = \Theta(n^2)$.

2. $T(n) = T(2n/3) + 1$

in which $a = 1$, $b = 3/2$, $f(n) = 1$, and $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$. Case 2 applies, since $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, and thus the solution to the recurrence is $T(n) = \Theta(\lg n)$.

3. $T(n) = 3T(n/4) + n \lg n$

we have $a = 3$, $b = 4$, $f(n) = n \lg n$, and $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$. Since $f(n) = \Omega(n^{\log_4 3 + \epsilon})$, where $\epsilon \approx 0.2$, case 3 applies if we can show that the regularity condition holds for $f(n)$. For sufficiently large n, we have that $af(n/b) = 3(n/4)\lg(n/4) \le (3/4)n \lg n = cf(n)$ for $c = 3/4$. Consequently, by case 3, the solution to the recurrence is $T(n) = \Theta(n \lg n)$.

4. $T(n) = 2T(n/2) + 2^n$: Master's Theorem Not Applicable

Thank You