Lecture 11

Mathematical Analysis of Recursive Algorithms and Solving Recurrences: Recursive Tree Method, Master Theorem for Solving Recurrences.

The recursion-tree method

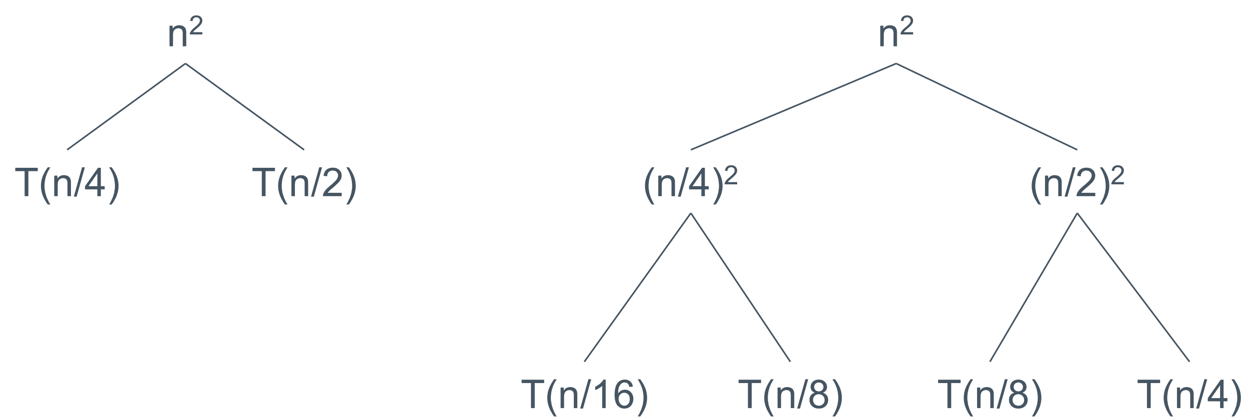
Convert the recurrence into a tree:

- *Each node represents the cost incurred at various levels of recursion*
- *Sum up the costs of all levels*

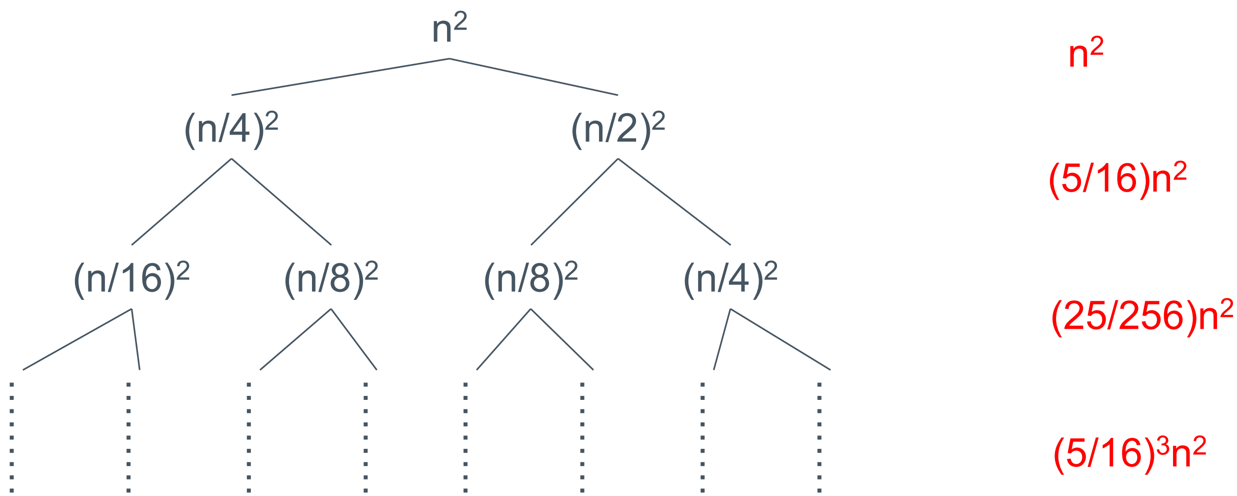
Used to "guess" a solution for the recurrence

Recursion tree

- *Visualizing recursive tree method*
- *eg.* $T(n)=T(n/4)+T(n/2)+n^2$



Recursion tree (Cont !!!)



When the summation is infinite and $|x| < 1$, we have the infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}, \qquad (3.4)$$

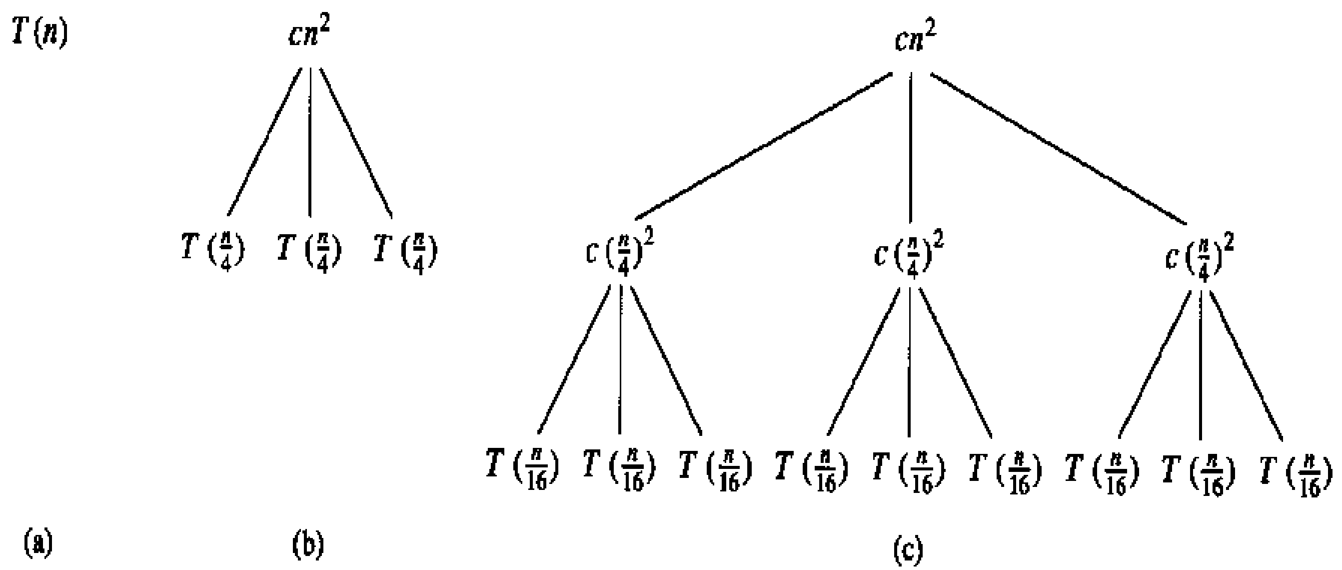
$$n^2 + \frac{5}{16}n^2 + \frac{25}{256}n^2 + \left(\frac{5}{16}\right)^3 n^2 + \ldots$$

$$= n^2 + \left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \ldots\right)$$

$$\cong n^2 \cdot \frac{1}{1 - 5/16} = \Theta\left(n^2\right)$$

Recursion-tree method (Cont !!!)

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$

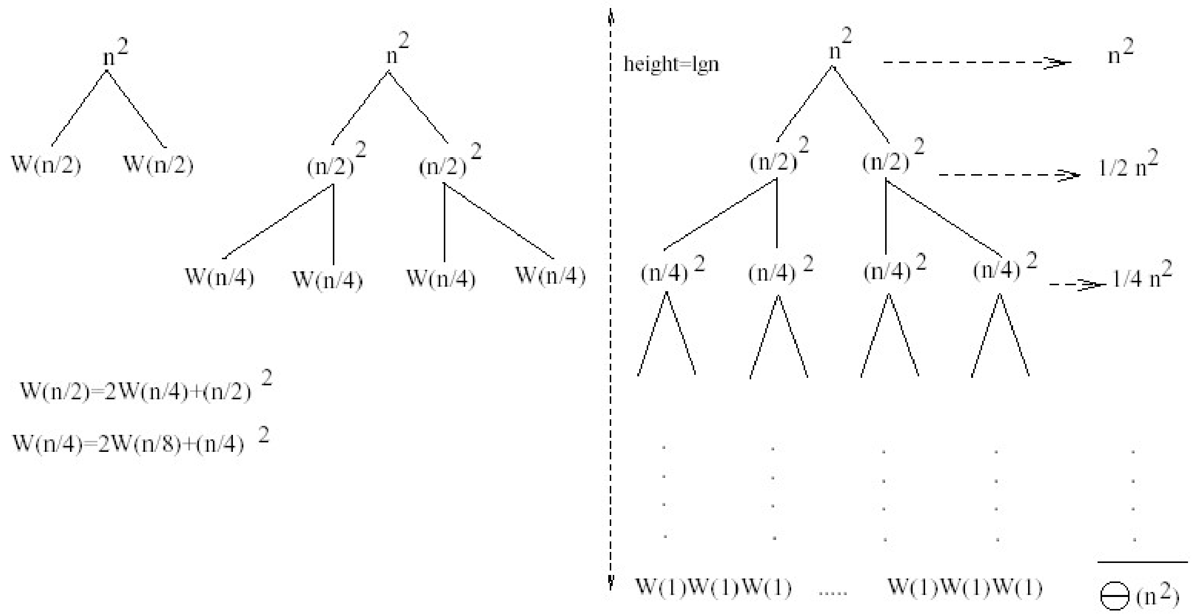


Recursion-tree method (Cont !!!)

- *Subproblem size for a node at depth i* $\dfrac{n}{4^i}$
- *Total level of tree* $\log_4 n + 1$
- *Number of nodes at depth i* 3^i
- *Cost of each node at depth i* $c(\dfrac{n}{4^i})^2$
- *Total cost at depth i* $3^i c(\dfrac{n}{4^i})^2 = (\dfrac{3}{16})^i cn^2$
- *Last level, depth* $\log_4 n$ *, has* $3^{\log_4 n} = n^{\log_4 3}$ *nodes*

Example

$W(n) = 2W(n/2) + n^2$



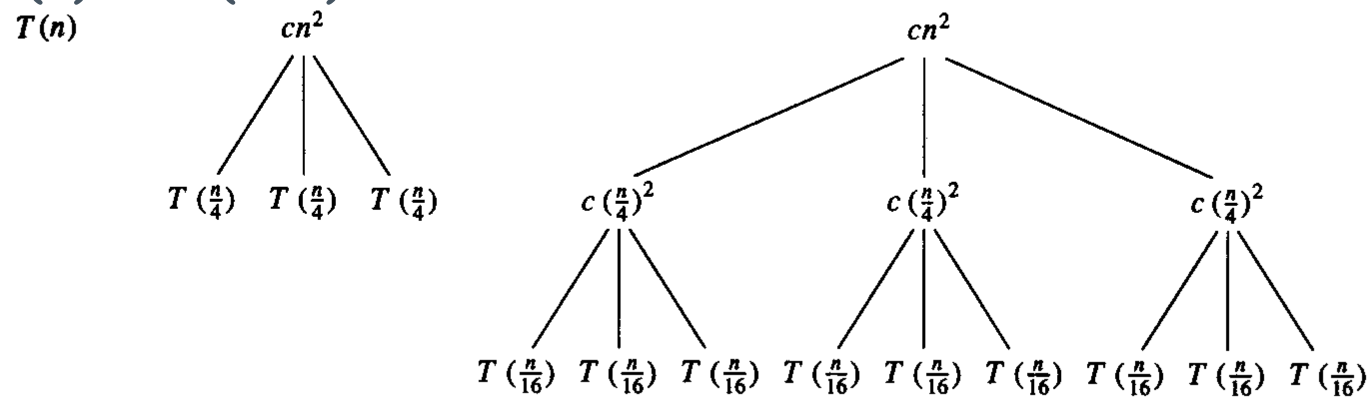
- *Subproblem size at level i is:* $n/2^i$
- *Subproblem size hits 1 when* $1 = n/2^i \Rightarrow i = \lg n$
- *Cost of the problem at level i* $= (n/2^i)^2$ *No. of nodes at level i* $= 2^i$
- *Total cost:*

$$W(n) = \sum_{i=0}^{\lg n - 1} \frac{n^2}{2^i} + 2^{\lg n} W(1) = n^2 \sum_{i=0}^{\lg n - 1} \left(\frac{1}{2}\right)^i + n \le n^2 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i + O(n) = n^2 \frac{1}{1 - \frac{1}{2}} + O(n) = 2n^2$$

$\Rightarrow W(n) = O(n^2)$

Example

E.g.: $T(n) = 3T(n/4) + cn^2$

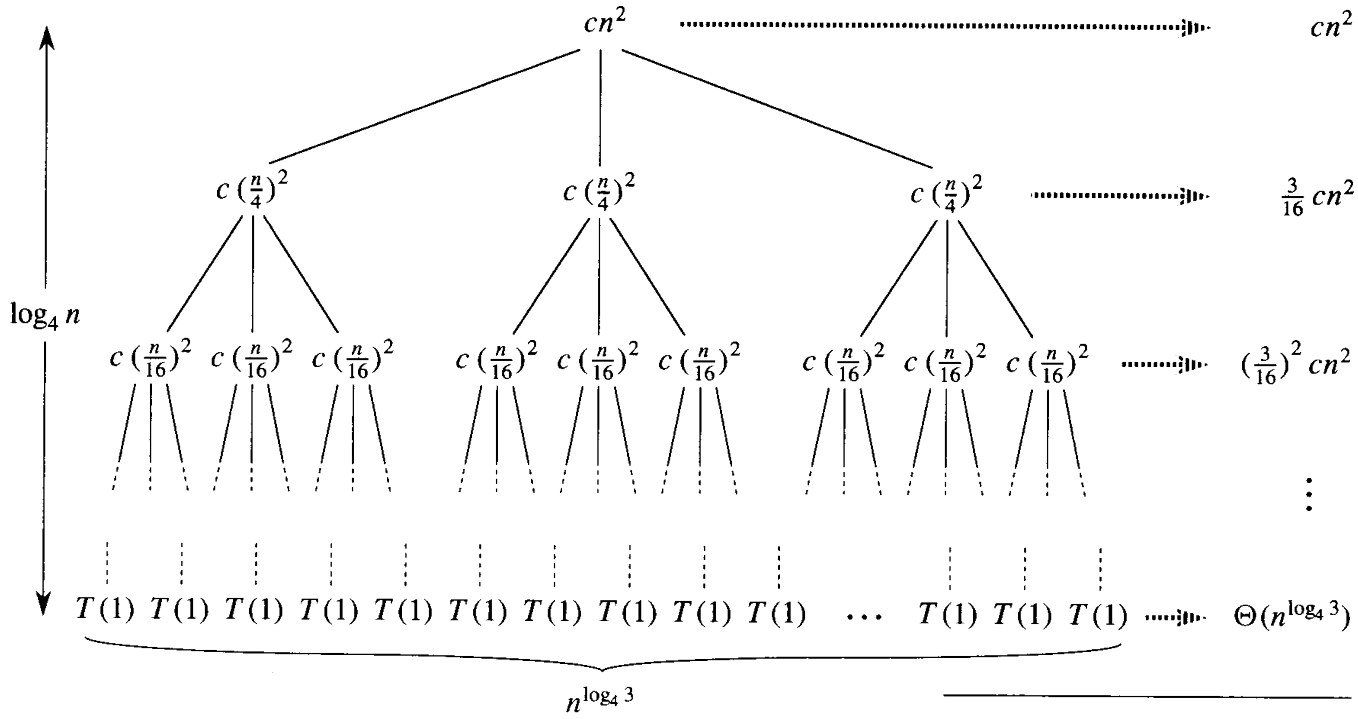


- *Subproblem size at level i is:* $n/4^i$
- *Subproblem size hits 1 when* $1 = n/4^i \Rightarrow i = \log_4 n$
- *Cost of a node at level i* $= c(n/4^i)^2$
- *Number of nodes at level i* $= 3^i \Rightarrow$ *last level has* $3^{\log_4 n} = n^{\log_4 3}$ *nodes*
- *Total cost:*

$$T(n) = \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) \le \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{1}{1-\frac{3}{16}} cn^2 + \Theta\left(n^{\log_4 3}\right) = O(n^2)$$

$\Rightarrow T(n) = O(n^2)$

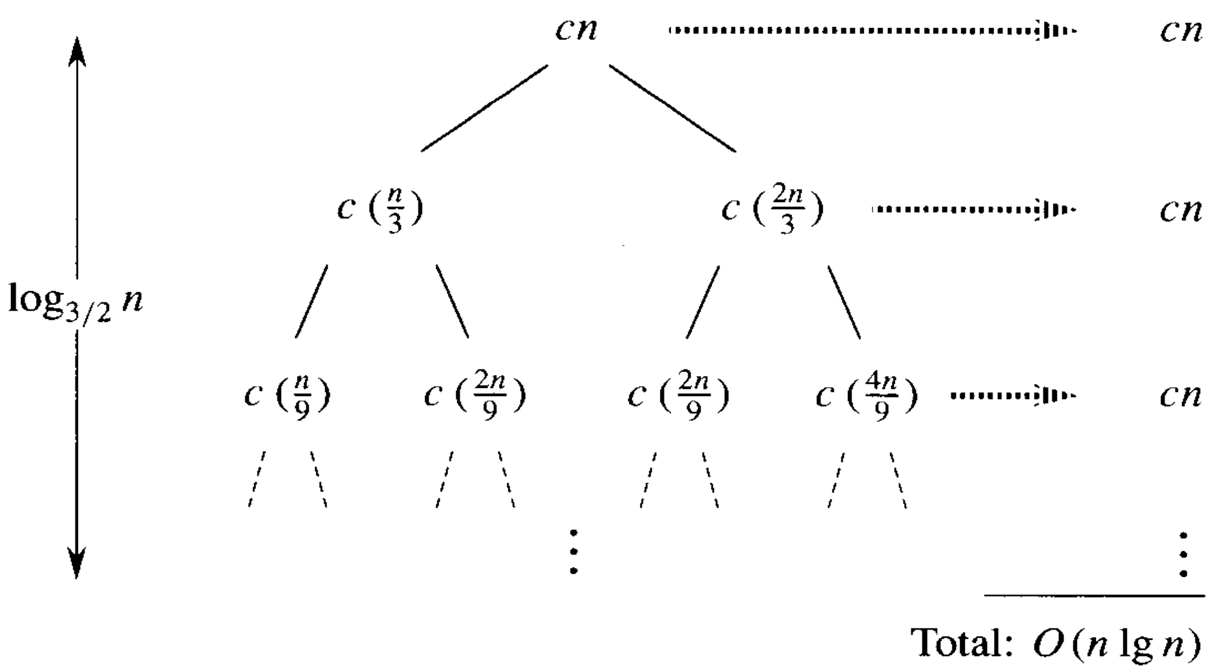
Recursion-tree method (Cont !!!)



(d)

Recursion-tree method((Cont !!!)

$$T(n) = T(n/3) + T(2n/3) + cn$$



Explain the Master Method

- A utility method for analysing recurrence relations

- Useful in many cases for divide and conquer algorithms

- These recurrence relations are of the form:

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

with $a >= 1$
and $b > 1$

- n = the size of the current problem
- a = the number of subproblems in the recursion
- n/b = the size of each subproblem
- f(n) = the cost of the work that has to be done outside the recursive calls (cost of dividing + merging)

Explain the Master Method

The cases

There are 3 cases:

1. The running time is dominated by the cost at the leaves:

$$\text{If } f(n) = O(n^{\log_b(a) - \varepsilon}), \text{ then } T(n) = \Theta(n^{\log_b(a)})$$

for an $\varepsilon > 0$

2. The running time is evenly distributed throughout the tree:

$$\text{If } f(n) = \Theta(n^{\log_b(a)}), \text{ then } T(n) = \Theta(n^{\log_b(a)} \log(n))$$

3. The running time is dominated by the cost at the root:

$$\text{If } f(n) = \Omega(n^{\log_b(a) + \varepsilon}), \text{ then } T(n) = \Theta(f(n))$$

for an $\varepsilon > 0$

If $f(n)$ satisfies the regularity condition:
$af(n/b) <= cf(n)$ where $c < 1$ (this always holds for polynomials)
Because of this condition, the Master Method cannot solve every recurrence of the given form.

How to apply the Master Method (step-by-step)

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

1. Extract a, b and f(n) from a given recurrence.

2. Determine $n^{\log_b(a)}$.

3. Compare f(n) and $n^{\log_b(a)}$ asymptotically.

4. Determine the appropriate Master Method case and apply it.

Example 1

Imagine that: $T(n) = 2T(n/2) + n$.

1. Extract; $a = 2$, $b = 2$ and $f(n) = n$.
2. Determine; $n^{\log_b(a)} = n^{\log_2(2)} = n^1 = n$.
3. Compare; $n^{\log_b(a)} = n$
   $f(n) = n$ → $=$
4. Thus case 2; evenly distributed

$$\text{Because } f(n) = \Theta(n),$$

$$\begin{aligned} T(n) &= \Theta(n^{\log_b(a)}\log(n)) \\ &= \Theta(n^1\log(n)) \\ &= \Theta(n\log(n)) \end{aligned}$$

Example 2

Imagine that: $T(n) = 9T(n/3) + n$.

1. Extract; $a = 9$, $b = 3$ and $f(n) = n$.

2. Determine; $n^{\log_b(a)} = n^{\log_3(9)} = n^2$.

3. Compare; $n^{\log_b(a)} = n^2$

   $f(n) = n$ <

4. Thus case 1; (Express $f(n)$ in terms of $n^{\log_b(a)}$)

   Because $f(n) = O(n^{2-\varepsilon})$,

   $T(n) = \Theta(n^{\log_b(a)}) = \Theta(n^2)$.

Example 3

Imagine that: $T(n) = 3T(n/4) + n\log(n)$.

1. Extract; $a = 3$, $b = 4$ and $f(n) = n\log(n)$.

2. Determine; $n^{\log_b(a)} = n^{\log_4(3)}$ where $\log_4(3) < 1$

3. Compare; $n^{\log_b(a)} = n^{\log_4(3)}$ $>$
   $f(n) = n\log(n)$

4. Thus case 3, but we have to check the regularity condition!

   The following should be true: $af(n/b) <= cf(n)$ where $c < 1$

   $\Leftrightarrow a(n/b)\log(n/b) <= cf(n)$ $\Leftrightarrow 3(n/4)\log(n/4) <= cf(n)$

   $\Leftrightarrow 3/4\ n\log(n/4) <= cf(n)$, this is true for $c = 3/4$, for example. ✔

   So because $f(n) = \Omega(n^{\log_4(3) + \varepsilon})$,

   $$T(n) = \Theta(f(n)) = \Theta(n\log(n))$$

Analysis

Further Explanation

- *There are four methods to solve a recursive relation*
  - *Iterative*
    - *In iterative method you will Convert the recurrence into a summation and try to bound it using known series.*
  - *Substitution*
    - *In substitution method, you will use guess or induction process to solve a recursive relation*
  - *Tree method*
    - *In Tree method, you will form a tree and then sum up the values of nodes and also use guesses*
  - **Master Theorem**:
    - *Only Specific problems can be solved in the form if recurrence relation is in the format like* $T(n) = aT\left(\frac{n}{b}\right) + f(n)\ where\ a \geq 1\ and\ b > 1$

Master Theorem

- *Let T(n) be a monotonically increasing function that satisfies*

$$T(n) = a\,T(n/b) + f(n)$$

$$T(1) = c$$

where $a \geq 1$, $b \geq 2$ or $b>1$, $c>0$. If $f(n)$ is $\Theta(n^d)$ where $d \geq 0$ then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Master Theorem: Example 1

- *Let* $T(n) = T(n/2) + \frac{1}{2} n^2 + n$. *What are the parameters?*

$a = 1$

$b = 2$

$d = 2$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$1 < 2^2$, *case 1 applies*

- *We conclude that* $T(n) \in \Theta(n^d) = \Theta(n^2)$

Master Theorem: Example 2

- *Let $T(n) = 2\,T(n/4) + \sqrt{n} + 42$. What are the parameters?*

$a =$ 2

$b =$ 4

$d =$ 1/2

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$2 = 4^{1/2}$, case 2 applies

- *We conclude that*

$$T(n) \in \Theta(n^d \log n) = \Theta(\log n \sqrt{n})$$

Master Theorem: Example 3

- *Let T(n)= 3 T(n/2) + 3/4n + 1. What are the parameters?*

$a =$ 3

$b =$ 2

$d =$ 1

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$3 > 2^1$, case 3 applies

- *We conclude that*

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3})$$

- *Note that $\log_2 3 \approx 1.584\ldots$, can we say that $T(n) \in \Theta(n^{1.584})$*

No, because $\log_2 3 \approx 1.5849\ldots$ and $n^{1.584} \notin \Theta(n^{1.5849})$

Master Theorem: Example 4

- *Let T(n)= 2T(n/2) + n log n. What are the parameters?*

$a =$ 2

$b =$ 2

$d =$ 1

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$2 = 2^1$, *case 2 applies*

- *We conclude that*

$$T(n) = \Theta\ (n^1 \log n) = \Theta\ (n\log n)$$

Master Theorem: Example 5

- *Let T(n)= T(n/3) + n log n. What are the parameters?*

$a =$ 1

$b =$ 3

$d =$ 1

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$1 < 3^1$, *case 1 applies*

- *We conclude that*

$$T(n) = \Theta(n^1) = \Theta(n)$$

Master Theorem: Example 6

- *Let T(n)= 4T(n/2) + n. What are the parameters?*

$a =$ 4

$b =$ 2

$d =$ 1

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$4 > 2^1$, *case 3 applies*

- *We conclude that*

$$T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 4}\right) = \Theta(n^2)$$

Master Theorem: Example 7

- *Let $T(n) = 8T(n/2) + n^2$. What are the parameters?*

$a =$ 8

$b =$ 2

$d =$ 2

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$8 > 2^2$, case 3 applies

- *We conclude that*

$$T(n) = \Theta(n^{\log_2 8}) = \Theta(n^{\log_2 2^3}) = \Theta(n^3)$$

Master Theorem: Example 8

- *Let $T(n) = 9T(n/3) + n^3$. What are the parameters?*

$a = 9$

$b = 3$

$d = 3$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$9 = 3^3$, *case 2 applies*

- *We conclude that*

$$T(n) = \Theta(n^3 \log n) = \Theta(n^3 \log n)$$

Master Theorem: Example 9

- *Let $T(n) = T(n/2) + 1$. What are the parameters?*

  $a = 1$

  $b = 2$

  $d = 0$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$1 = 2^0$*, case 2 applies*

- *We conclude that*

$$T(n) = \Theta\,(n^0 \log n) = \Theta\,(\log n)$$

Recurrence Relation

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn & \text{if } n \geq 2 \end{cases}$$

Iterative Substitution

- *In the iterative substitution, or "plug-and-chug," technique, we iteratively apply the recurrence equation to itself and see if we can find a pattern*:

$$
\begin{aligned}
T(n) &= 2T(n/2) + bn \\
&= 2(2T(n/2^2)) + b(n/2)) + bn \\
&= 2^2 T(n/2^2) + 2bn \\
&= 2^3 T(n/2^3) + 3bn \\
&= 2^4 T(n/2^4) + 4bn \\
&= \ldots \\
&= 2^i T(n/2^i) + ibn
\end{aligned}
$$

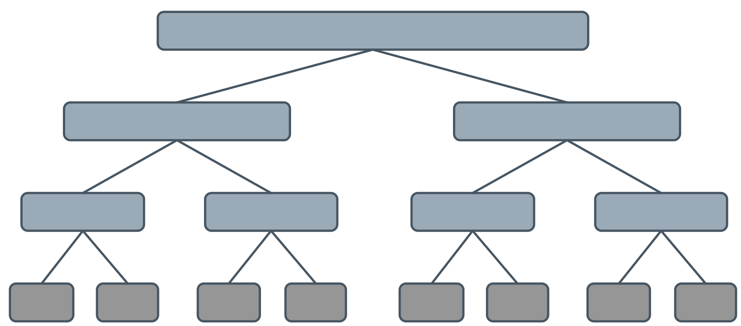
- *Note that base, T(n)=b, case occurs when 2^i=n. That is, i = log n. So,* $T(n) = bn + bn\log n$
- *Thus, T(n) is O(n log n).*

The Recursion Tree

- *Draw the recursion tree for the recurrence relation and look for a pattern:*

$$T(n) = \begin{cases} b & \text{if } n < 2 \\ 2T(n/2) + bn & \text{if } n \geq 2 \end{cases}$$

depth	T's	size	time
0	1	n	bn
1	2	$n/2$	bn
i	2^i	$n/2^i$	bn
…	…	…	…



Total time = $bn + bn \log n$

Master Theorem:

- *Let T(n)= 2T(n/2) + bnlogn. What are the parameters?*

$a = 2$

$b = 2$

$d = 1$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Therefore, which condition applies?

$2 = 2^1$, *case 2 applies*

- *We conclude that*

$$T(n) = \Theta\ (n^1 \log n) = \Theta\ (n \log n)$$

Summary

- *Let T(n) be a monotonically increasing function that satisfies*

$$T(n) = a\,T(n/b) + f(n)$$

$$T(1) = c$$

where $a \geq 1$, $b \geq 2$, $c>0$. *If* $f(n)$ *is* $\Theta(n^d)$ *where* $d \geq 0$ *then*

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{If } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Thank You!!!

Have a good day