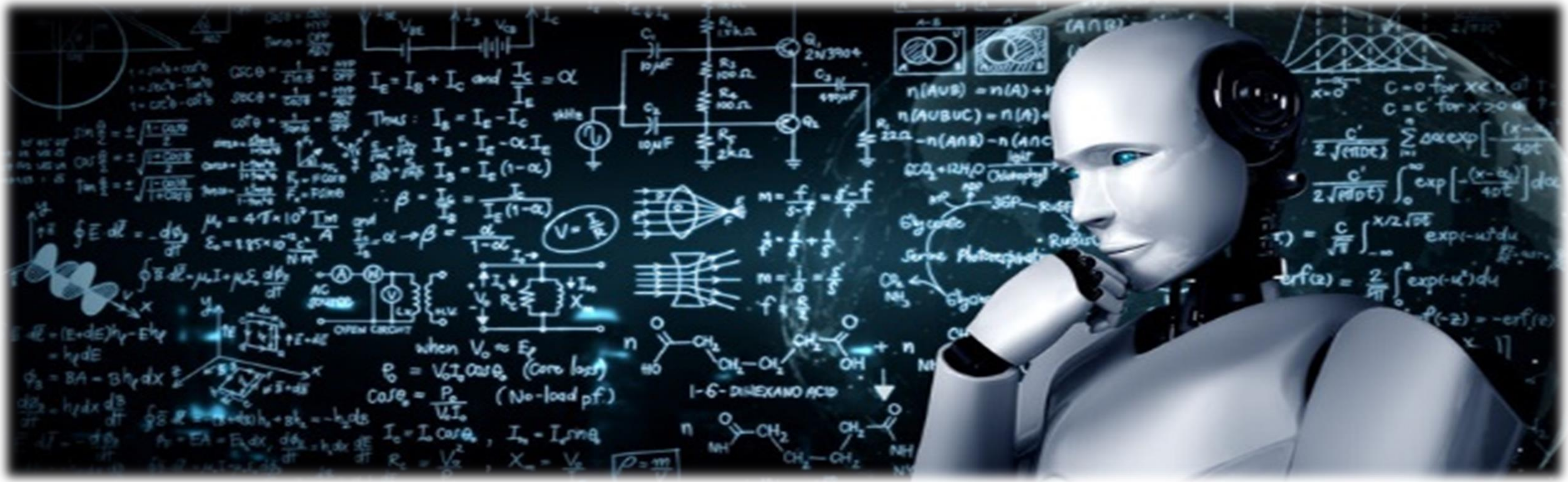


Lecture 04 - Introduction to Algorithms Recursion – Part 2

Design and Analysis of Algorithms – IT1205

Year 02 Semester 02

SCU
SLIIT CITY UNI

Contents for today

- Finding a solution to a recurrence
  - Repeated Substitution method.
  - Recursion tree.
  - Master Theorem.

The Recursion-Tree Method

- In a **Recursion Tree**, each node represents the cost of a single sub problem somewhere in the set of recursive function invocations.

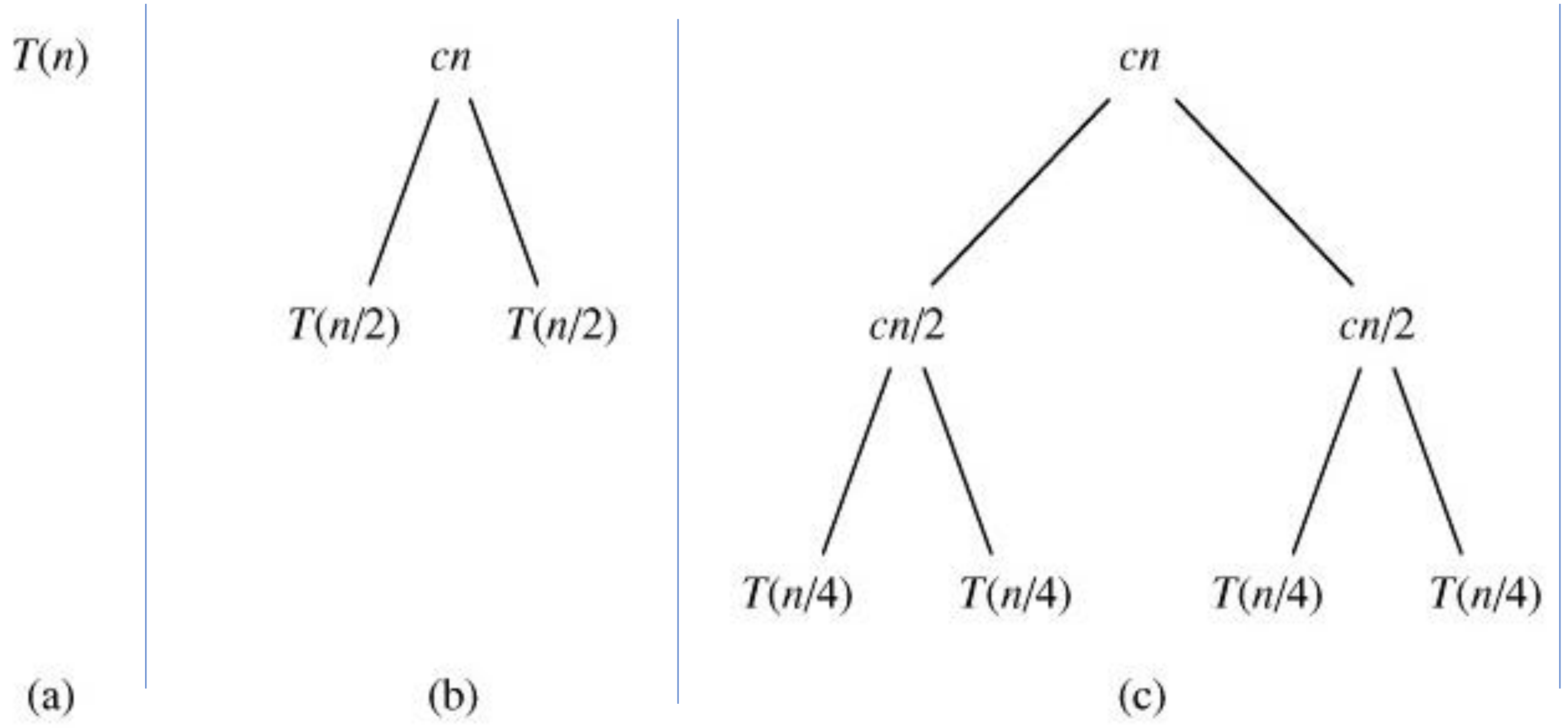
Steps in Recursion Tree Method

- Draw a recursive tree for given recurrence relation
- Calculate the cost at each level and count the total no of levels in the recursion tree.
- Count the total number of nodes in the last level and calculate the cost of the last level
- Sum up the cost of all the levels in the recursive tree

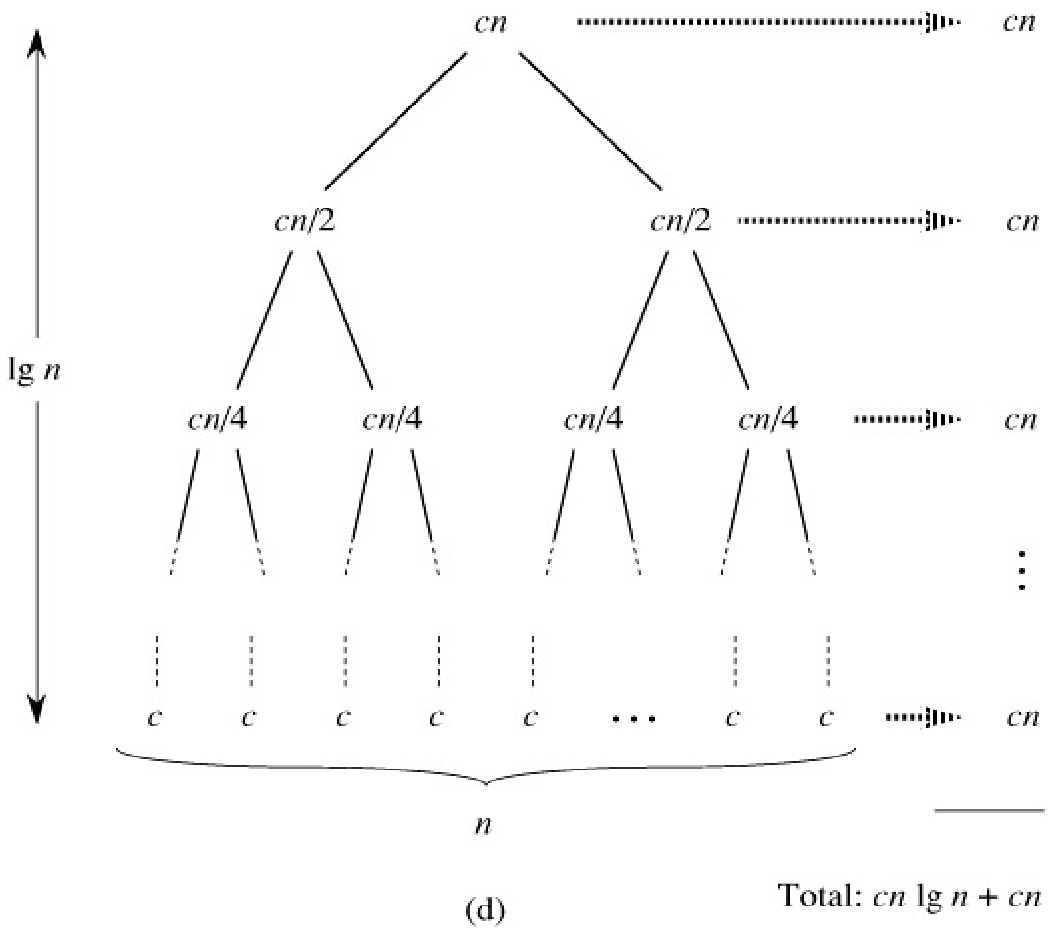
Examples

Let's try the Recursion Tree method

Recursion tree for $T(n) = 2T(n/2) + cn$



Recursion tree for $T(n) = 2T(n/2) + cn$



(d)

The Master Method

- The Master method applies to recurrences of the form

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

where $a \geq 1$ and $b > 1$, and $f(n)$ is an asymptotically positive function.

The recurrence describes the running time of an algorithm that divides a problem of size n into a sub problems, each of size n/b, where a and b are positive constants. The a sub problems are solved recursively, each in time T (n/b). The cost of dividing the problem and combining the results of the sub problems is described by the function f (n).

The master theorem

- The master method depends on the following 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 $T(n)$ can be bounded asymptotically as follows.

The Master Theorem

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

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$.
($f(n)$ is polynomially smaller than $n^{\log_b a}$.)
Solution: $T(n) = \Theta(n^{\log_b a})$.

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$.

($f(n)$ is within a polylog factor of $n^{\log_b a}$, but not smaller.)
Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
(Intuitively: cost is $n^{\log_b a} \lg^k n$ at each level, and there are $\Theta(\lg n)$ levels.)
Simple case: $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$.

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $f(n)$ satisfies the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n.
($f(n)$ is polynomially greater than $n^{\log_b a}$.)
Solution: $T(n) = \Theta(f(n))$.
(Intuitively: cost is dominated by root.)

The master theorem

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a-\varepsilon}\right) \to f(n) < n^{\log_b a} \\ \Theta\left(n^{\log_b a} \lg n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \to f(n) = n^{\log_b a} \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right) \to f(n) > n^{\log_b a} \\ & \text{if } af(n/b) \le cf(n) \text{ for } \mathrm{c} < 1 \text{ and large } n \end{cases}$$

Master Theorem

Give tight asymptotic bound for

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

Solution:

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

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

$$f(n) = O(n^{\log_3 9-\varepsilon}) \text{ for } \varepsilon = 1 \text{ or } f(n) < n^{\log_3 9} \rightarrow \text{case1}$$

$$\therefore T(n) = \Theta(n^2)$$

Master Theorem

Give tight asymptotic bound for

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

Solution:

a=1, b=3/2, and $f(n)$ = 1.

$$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$$

$$f(n) = \Theta(n^{\log_b a}) \text{ or } f(n) = n^{\log_b a} \rightarrow \text{case } 2$$

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

Master Theorem

- Give tight asymptotic bound for

$T(n) = 3T(n/4) + n \log n$

Solution:

a=3, b=4, and $f(n) = n \log n$

.

$$n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$$

$$f(n) = \Omega(n^{\log_4 3+\varepsilon}), \text{for } \varepsilon \approx 0.2 \text{ or } f(n) > n^{\log_4 3} \rightarrow \text{case } 3$$

$$Note: n \lg n \geq c.n^{\log_4 3}.n^{0.2}$$

$$\therefore T(n) = \Theta(n.\log n)$$


SEE YOU
next
week!