Overview

- **Solving Recurrence relation**
  - **Recursion Tree**
  - **Master Method**
  - **Substitution Method**

Running Time - Recurrence equation

- Describes the running time of recursive algorithms

  T(n) in terms of running times of smaller subproblems

- Solve the recurrence using mathematical tools to get bounds on the running time

Running Time-recurrence

- **Factorial:**

$$T(n) = T(n\text{-}1) + c \quad \text{if } n>1$$
$$= d \quad \text{if } n<=1$$

- **Binary Search:**

$$T(n) = T(n/2) + c \quad \text{if } n>1$$
$$= d \quad \text{if } n <= 1$$

- **Merge Sort:**

$$T(n) = 2T(n/2) + cn \quad \text{if } n>1$$
$$= d \quad \text{if } n <= 1$$

Divide and Conquer – Recurrence

$$T(n) = \Theta(1) \quad \textbf{if } n<=c$$

$$= a\,T(n/b) + D(n) + C(n) \quad \textbf{otherwise}$$

- Number of subproblems – ***a***
- Each subproblem size is ***1/b*** the size of the original
- ***D(n)*** – time to divide the problem into subproblems
- ***C(n)*** – time to combine the solutions

Divide and Conquer – Recurrence

$$
\begin{aligned}
T(n) &= \Theta(1) && \textbf{if } n<=c \\
&= a\,T(n/b) + f(n) && \textbf{otherwise} \quad (f(n) = D(n)+C(n))
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= d && \textbf{if } n<= 1 \\
&= 2\,T(n/2) + cn && \textbf{otherwise}
\end{aligned}
$$

$a = 2$, $b=2$, $f(n)=cn$ *(a, b need not be same)*

Solving recurrence

- **Substitution method**
  - **guess a solution, use mathematical induction to prove**
- **Recursion Tree**
  - **nodes represent cost. Sum up the cost at different levels**
- **Master Method**
  - **memorize three cases**

Merge Sort - Recurrence

Recurrence:

$$T(n) = \begin{cases} c & \textbf{if } n <= 1 \\ 2\,T(n/2) + cn & \textbf{otherwise} \end{cases}$$

Recurrence - Iteration

$T(n) \quad = 2\,T(n/2) + cn$

$= 2\,(2\,T(n/4) \;+ cn/2) + cn$

$= 4\,T(n/4) + \; 2cn$

$= 4\,(2\,T(n/8) + cn/4) + 2cn$

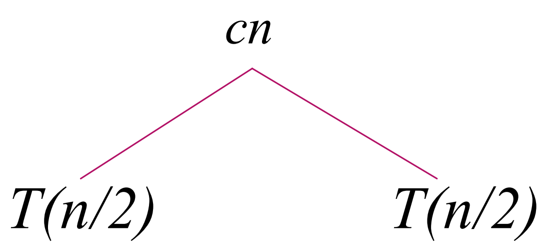
……..

Solving Recurrence – Recursion Tree

- **Recursion Tree** - Visualize the iteration of recurrence

  $T(n) = 2\,T(n/2) + cn$

T(n)



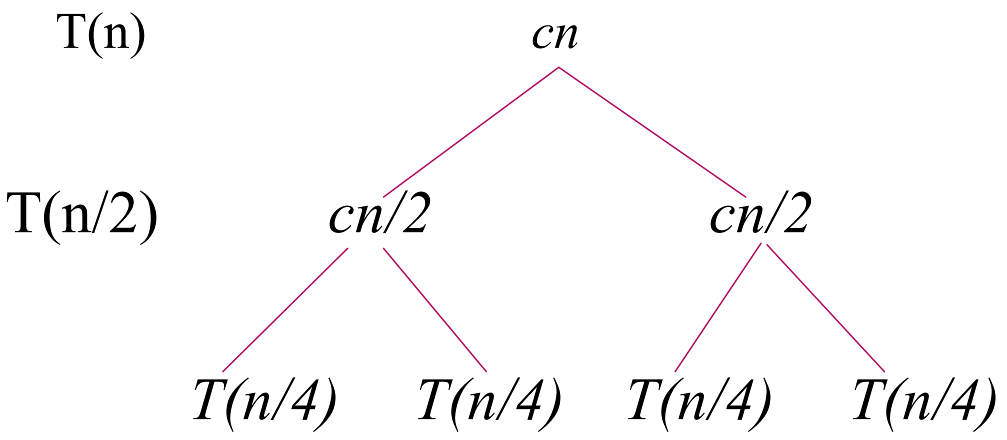
cn

$T(n/2)$ $T(n/2)$

assuming n is a power of 2

Recursion Tree

$$T(n) = 2\,T(n/2) + cn$$



$T(n)$

cn

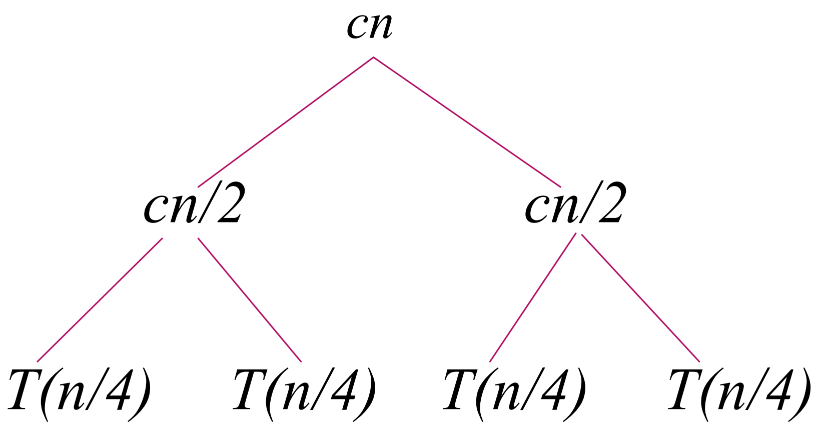
$T(n/2)$

$cn/2$ $cn/2$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

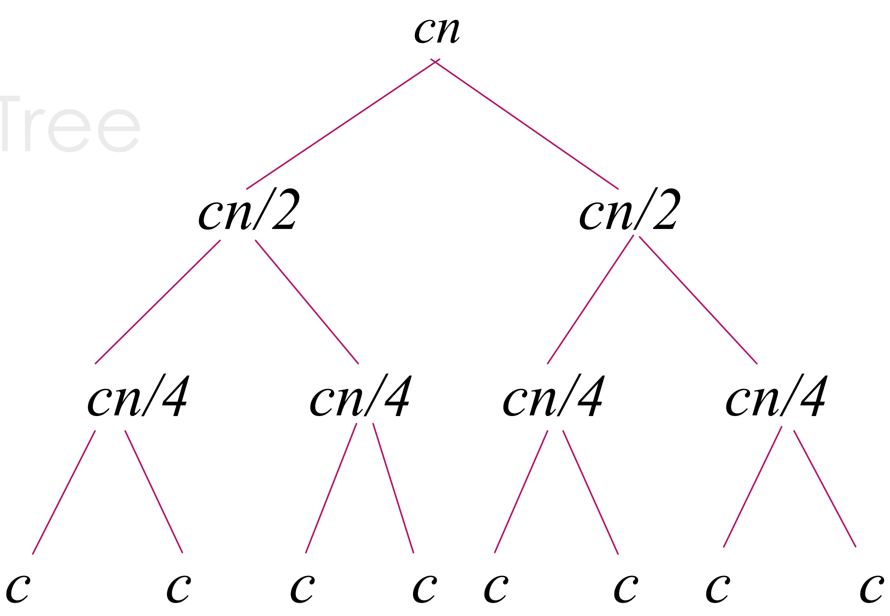
Recursions Tree

Trees in general - Node, Children, Root, Leaf



Recursion Tree

n=8

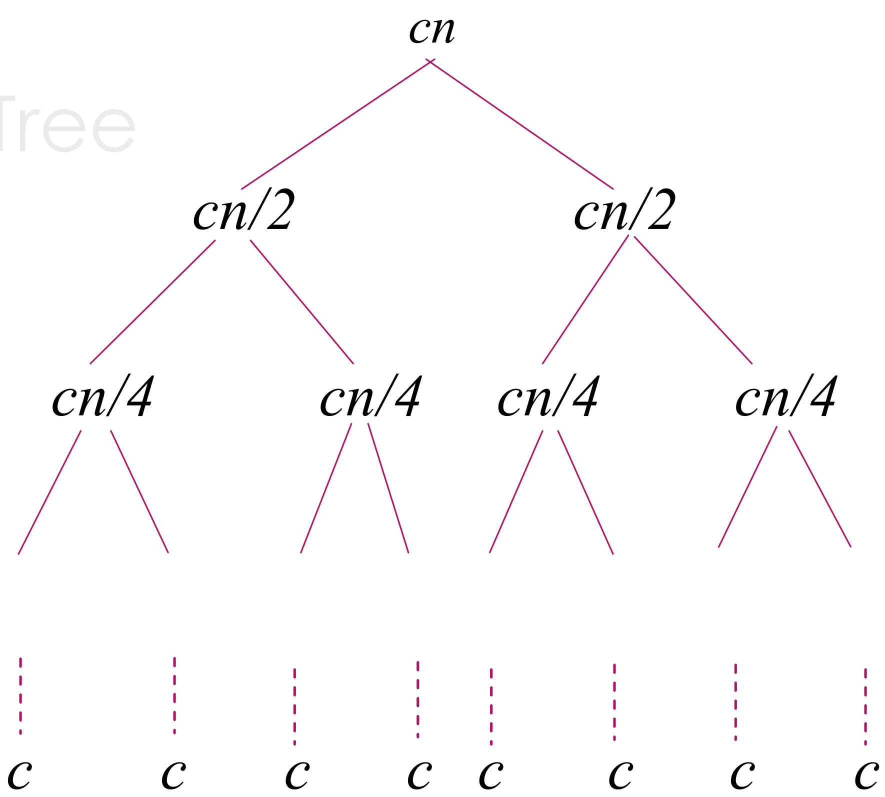


Number of levels in the tree = *4* (height = 3)

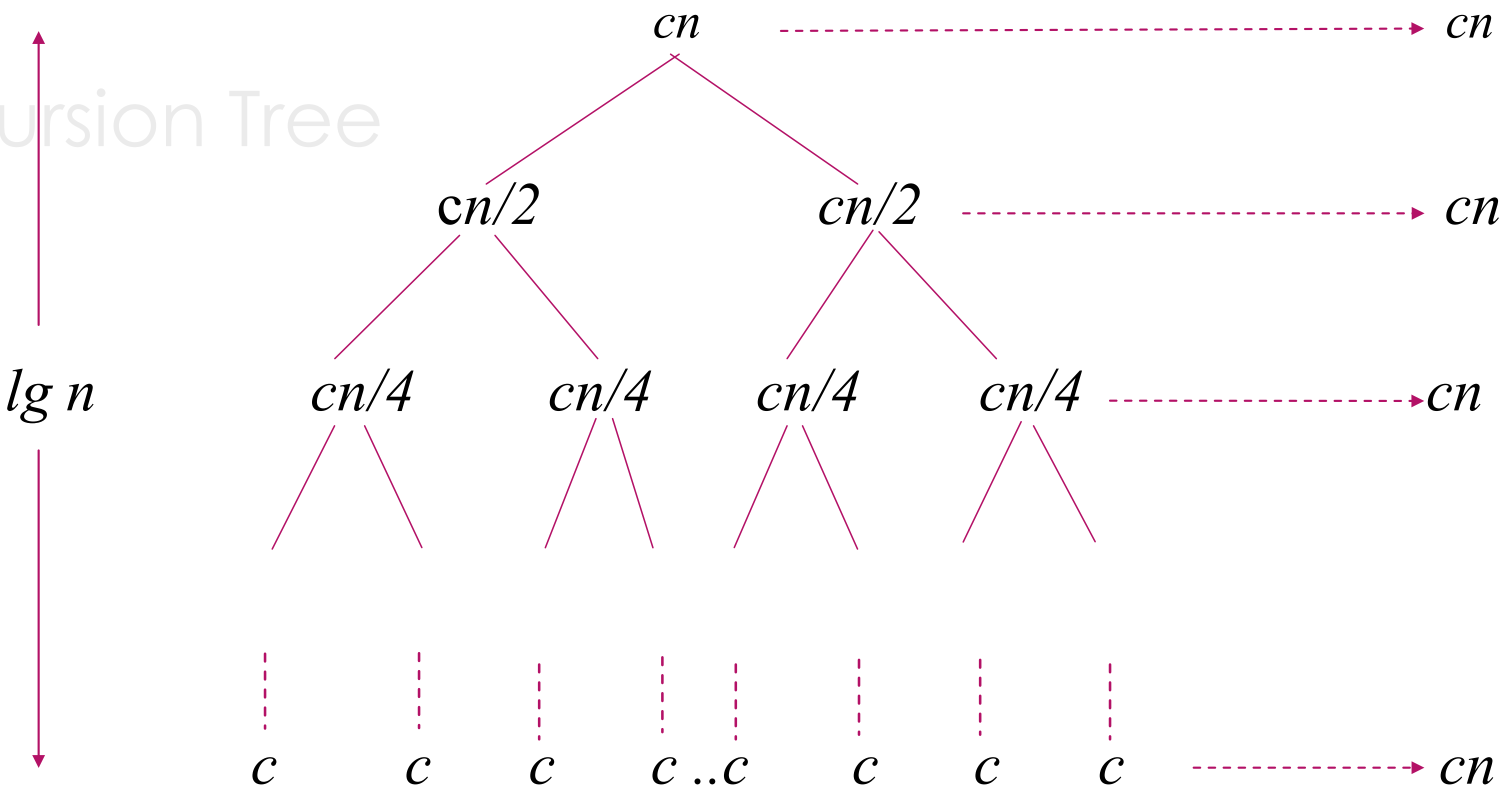
Sum of node cost in each level = *8c*

8 nodes at the leaf level each costing *c*

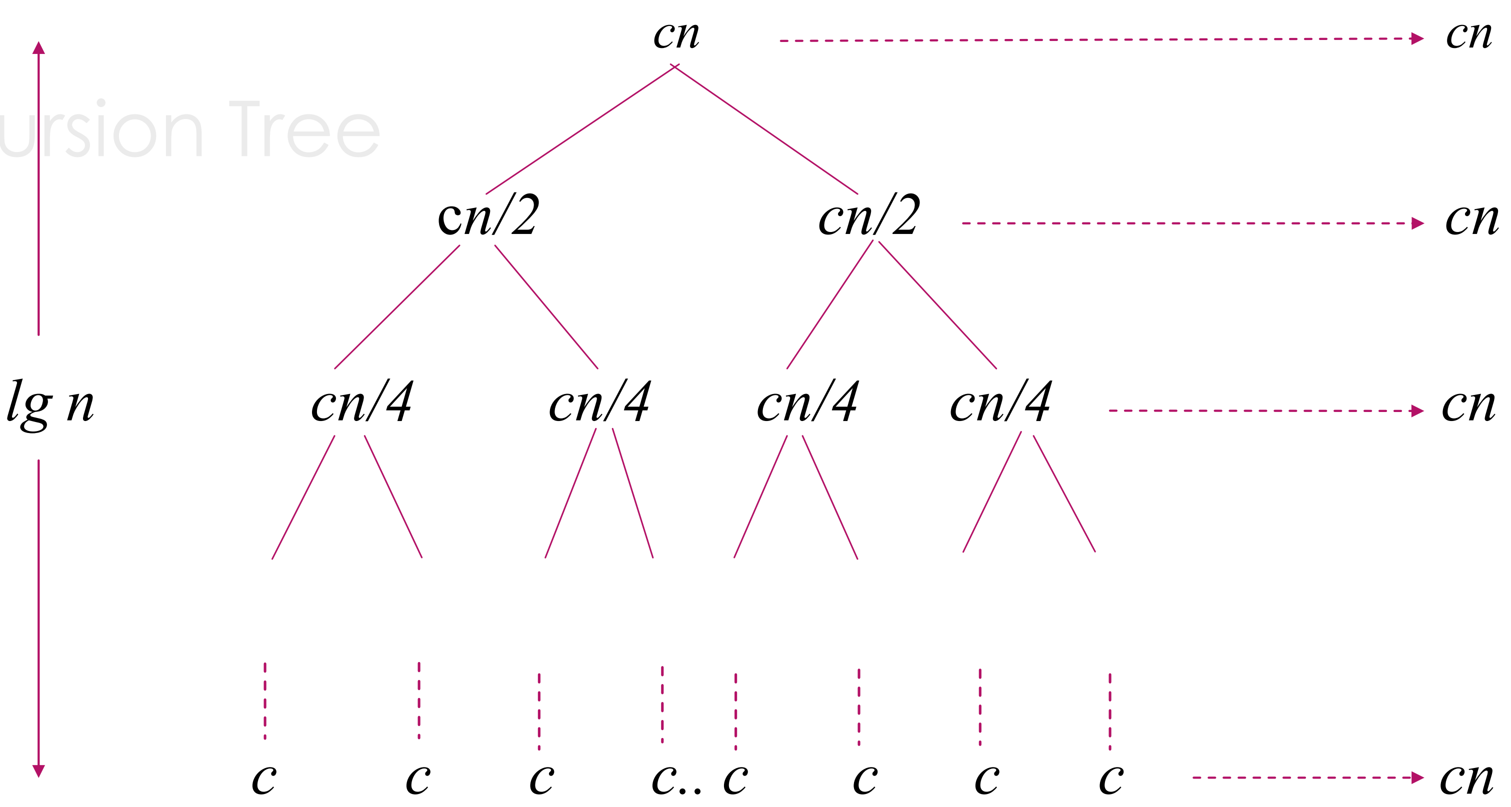
Recursion Tree



Recursion Tree



Recursion Tree



Number of Levels = *lg n +1* (Each level contributes total cost *cn)*

Total cost = *cn lg n* + *cn* **Running Time: *Θ(n lg n)***

Divide and Conquer – Recurrence

Solve using recursion tree:

$$T(n) = \begin{cases} d & \textbf{if } n<=1 \\ 2\,T(n/2) + c & \textbf{otherwise} \end{cases}$$

Divide and Conquer – Recurrence

Solve using recursion tree:

$$T(n) = \Theta(1) \qquad \text{if } n<=1$$

$$= 3\,T(n/4) + \Theta(n^2) \qquad \text{otherwise}$$

Divide and Conquer – Recurrence

Solve using recursion tree:

$$T(n) = \Theta(1) \qquad \text{if } n<=1$$

$$= 8\ T(n/2) + \Theta(n^2) \qquad \text{otherwise}$$

Reference

T H Cormen, C E Leiserson, R L Rivest, C Stein *Introduction to Algorithms*, 3rd ed., PHI, 2010