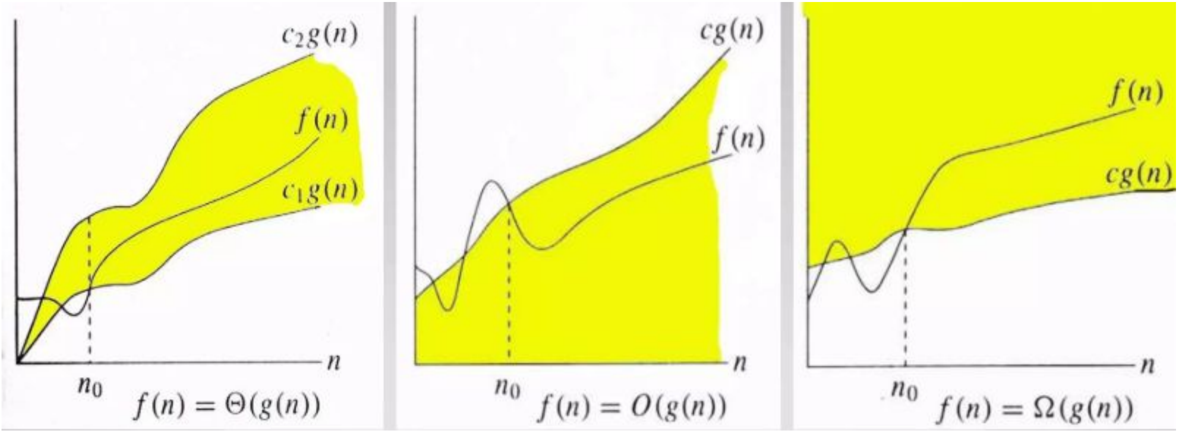
SCS1308 - Foundations of Algorithm

Tutorial - 02

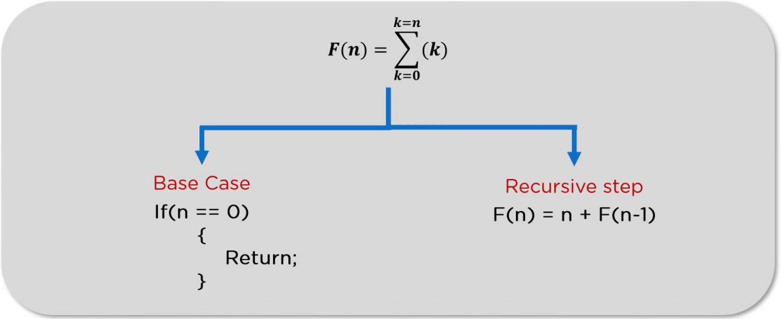
Time Complexity & Recursion Tree

Time complexity an algorithm

- Types of analysis
  - Worst case
  - Best case
  - Average case
- Comparisons often focus on growth rates (Big-O, Omega, Theta)



Recursion Basics



- ➔ Recursive algorithms consist of base cases and recursive cases.
- ➔ Whenever we analyze the run time of a recursive algorithm,
  - ○ We will first get a recurrence relation
  - ○ Then solve that recurrence relation
- ➔ **Recurrence relations express the overall time complexity.**

- ➔ T(n) =T(n-1)+n is an example of a recurrence relation
- ➔ A Recurrence Relation is any equation for a function T, where T appears on both the left and right sides of the equation.
- ➔ We always want to “solve” these recurrence relation by getting an equation for T, where T appears on just the left side of the equation

Methods to Solve Recurrences

1. Recursion Tree
2. Iteration Method
3. Substitution Method
4. Master's Theorem

Recursion Tree Method

Steps:

1. Build the tree

2. Compute TC per level

3. Compute number of levels

    (find last level as a function of N)

4. Compute total over levels.

    * Find closed form of that summation.

$$T(1) = c$$

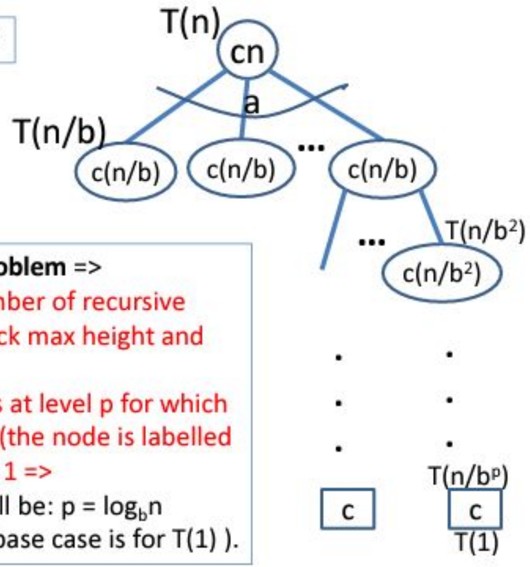
Problem size

The local TC at the node

$$T(n) = a*T(n/b) + cn$$

Number of subproblems =>
Number of children of a node in the recursion tree. =>
Affects the number of nodes per level. At level i there will be a^i nodes.
Affects the level TC.

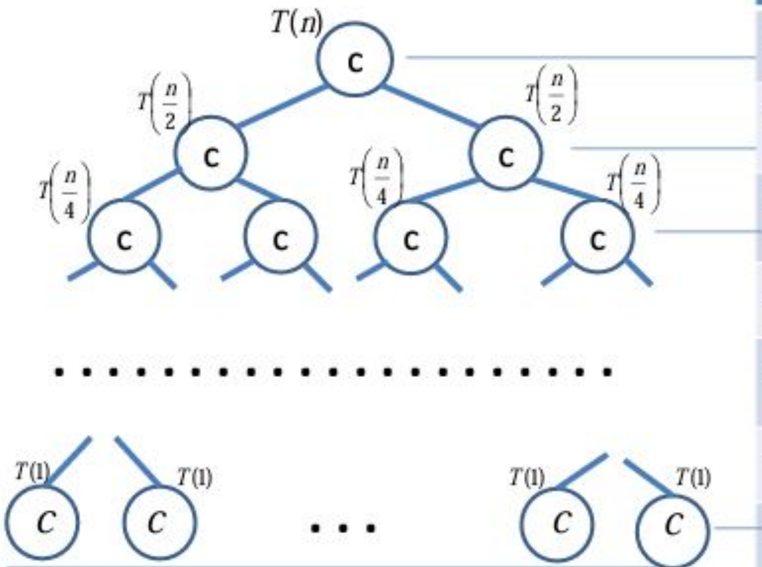
Size of a subproblem =>
Affects the number of recursive calls (frame stack max height and tree height)
Recursion stops at level p for which the pb size is 1 (the node is labelled T(1)) => $n/b^p = 1$ =>
Last level, p, will be: $p = \log_b n$
(assuming the base case is for T(1)).



TC = time complexity

Recursion Tree for: $T(n) = 2T(n/2)+c$

Base case: $T(1) = c$



Level	Arg/ pb size	TC of 1 node	Nodes per level	Level TC
0	n	c	1	c
1	n/2	c	2	2c
2	n/4	c	4	4c
…				
i	$n/2^i$	c	2^i	$2^i c$
…				
p=lgn	1 ($=n/2^p$)	c	2^p ($=n$)	$2^k c$

Stop at level p, when the subtree is T(1).
=> The problem size is 1, but the general formula for the problem size, at level p is:
$n/2^p$=> $n/2^p = 1$ => $p = \lg n$

Tree TC $= c(1+2+2^2+2^3+\ldots+2^i+\ldots+2^p)=c2^{p+1}/(2-1)$
$= 2c2^p = 2cn = \Theta(n)$

Thank you