Lecture 18 - Wednesday, March 15

Lecture

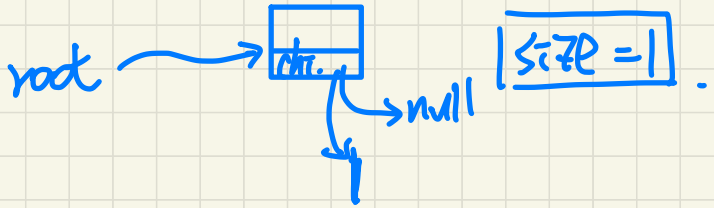
Binary Trees ADT

Definition, Terminology, Properties

Binary Trees: Recursive Definition

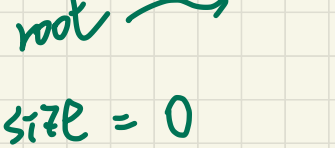
- root
- size

Case 1: A singleton tree

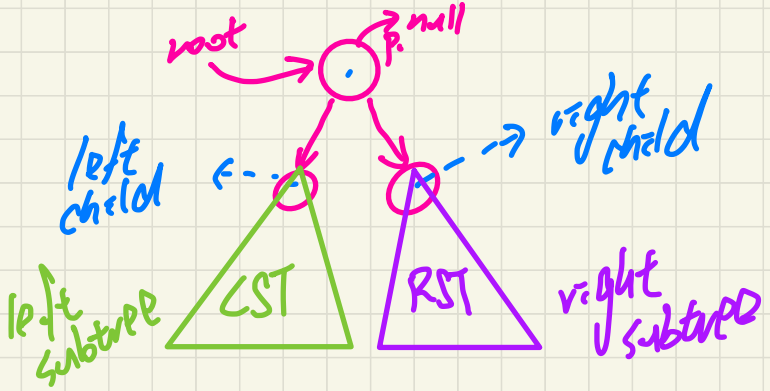


Case 0: Empty tree

ϕ

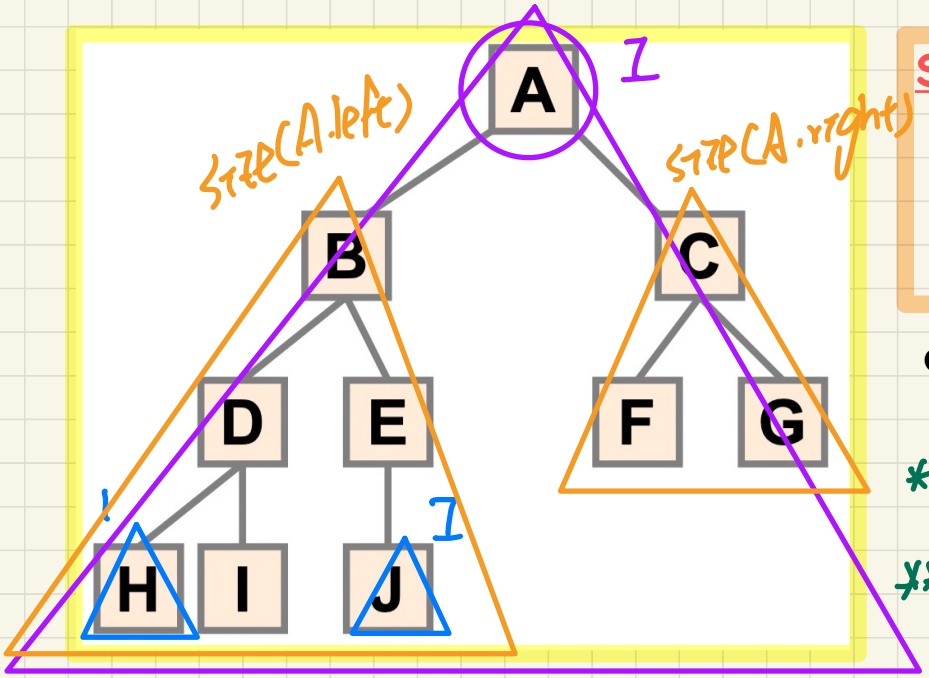


Case 2: ≥ 2 nodes



BT Terminology: LST vs. RST

search(target, int. n) = n.equals(t) || search(t, n.left)



Strategy of Recursion on BT:

+ Do something on root
+ Recur on LST || search(t, n.right)
+ Recur on RST

e.g.,

* + counting size

** + searching item

** search(target, ext. n) = n.equals(t)
e.g. T, F

* size(external n.) = 1
left right null

size(internal n.) = 1 + size(n.left) + size(n.right)
e.g. A

Deriving the Sum of a Geometric Sequence

Initial Term: I

Common Factor: r

Number of Terms: k

$I\ (1 + 2 + 4 + 8 + 16 + \cdots + 1024)$ — $1024 = 2^{10}$

$*2$, $*2$ — C.f.

$I \cdot r^0$

$$S_k = I + I\cdot r + I\cdot r^2 + I\cdot r^3 + \cdots + I\cdot r^{k-1}$$

$$r\cdot S_k = I\cdot r + I\cdot r^2 + I\cdot r^3 + \cdots + I\cdot r^{k-1} + I\cdot r^k$$

$$r\cdot S_k - S_k = (r-1)\cdot S_k = I\cdot r^k - I = I\cdot(r^k - 1)$$

$$S_k = \frac{I\cdot(r^k-1)}{r-1}$$

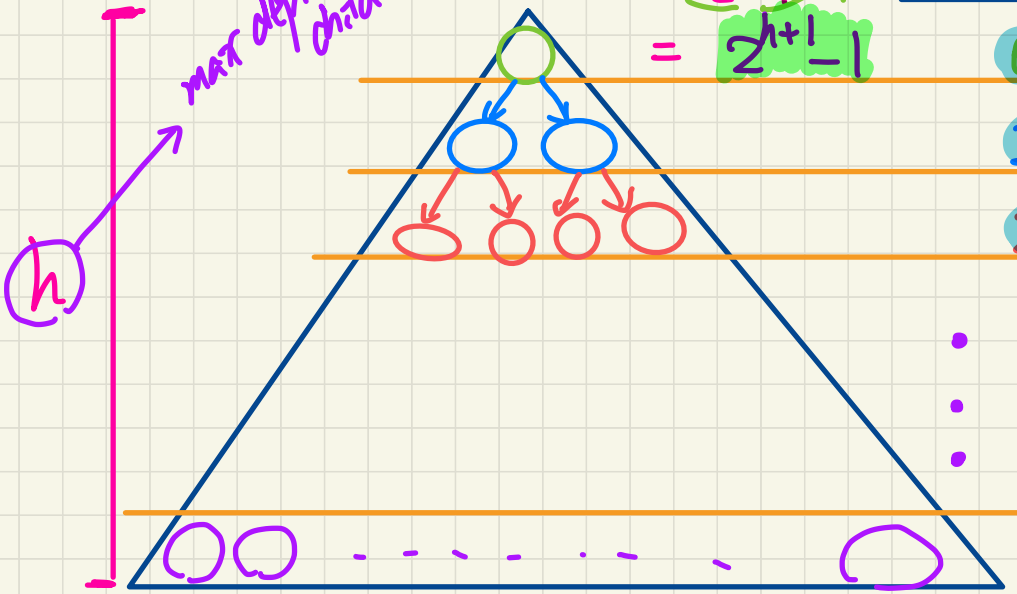
k

$$S_k = \frac{I\cdot(r^k-1)}{r-1}$$

BT Terminology: Depths, Levels, Max # of Nodes

Max # of nodes in a BT with height $h \geq 0$.

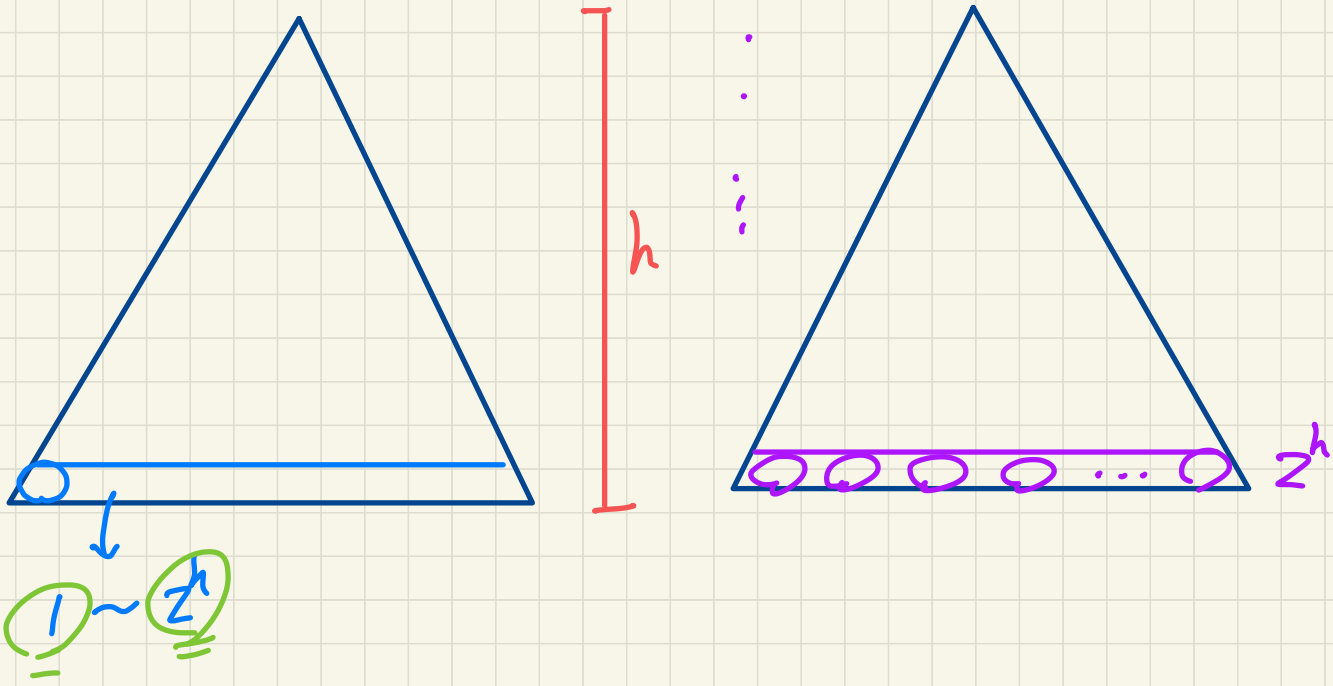
$$2^0 + 2^1 + 2^2 + \cdots + 2^h = \frac{2^0 (2^{h+1} - 1)}{2 - 1} = 2^{h+1} - 1$$



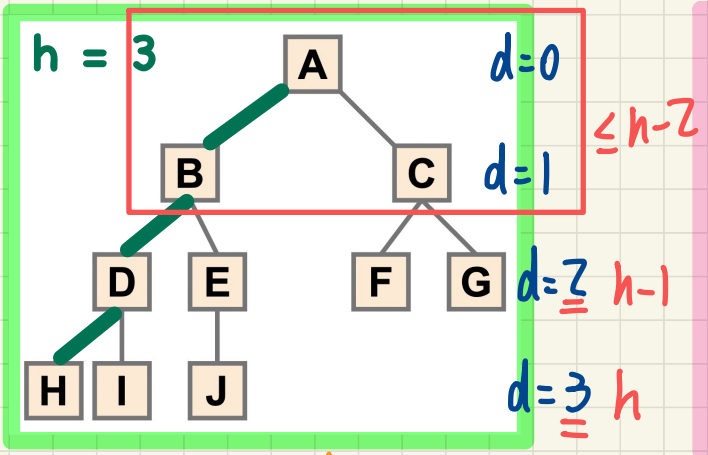
Level ? (d)	Max # Nodes at Level ?
0	$1 = 2^0$
1	$2 = 2^1$
2	$4 = 2^2$
⋮	⋮
h	? 2^h

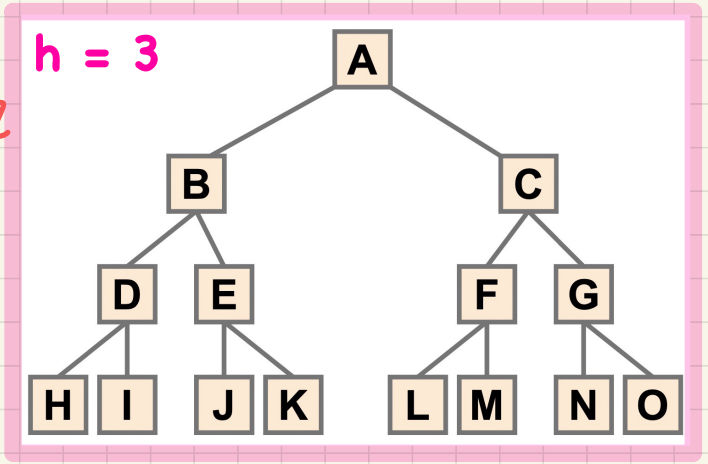
Exercise: Max # of nodes from Level 0 to Level h-1?

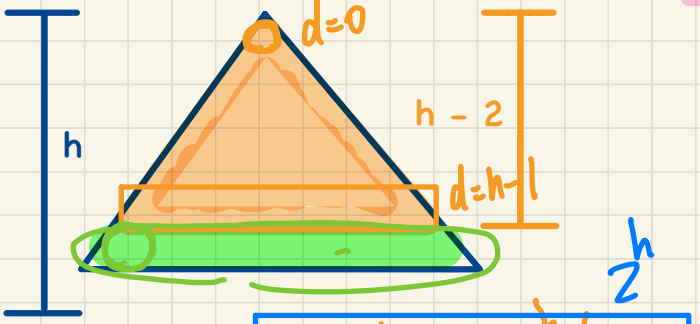
Complete BT vs. Full BT

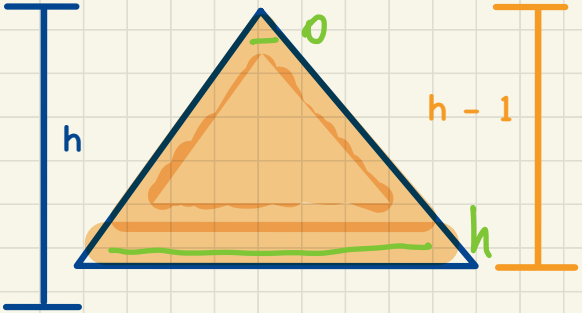


BT Terminology: Complete vs. Full BTs









Min # nodes? $(2^0 + 2^1 + \cdots + 2^{h-1}) + 1$

Max # nodes? $(2^0 + 2^1 + \cdots + 2^{h-1}) + 2^h$

$2^{h+1} - 1$

Min # nodes? $2^0 + 2^1 + \cdots + 2^h = 2^{h+1} - 1$

Max # nodes?

BT Properties: Bounding # of Nodes

Given a *binary tree* with ***height*** *h*, the ***number of nodes*** *n* is bounded as:

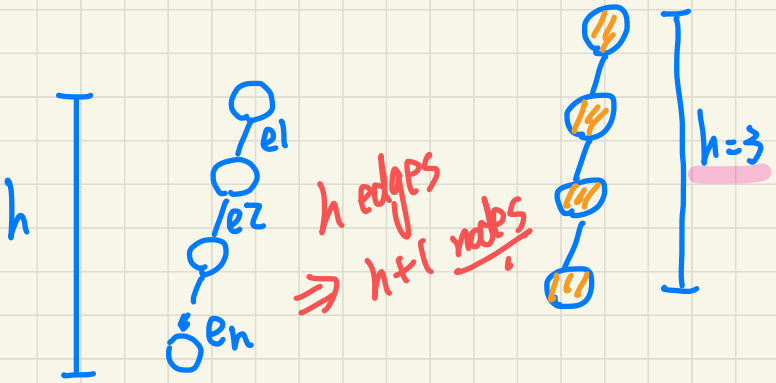
$$h + 1 \leq n \leq 2^{h+1} - 1$$

For example, say h = 3

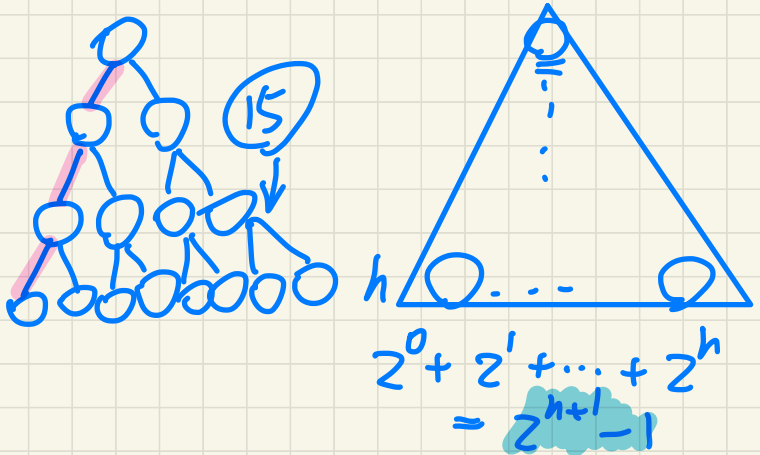
min: $3+1 = 4$

max: $2^{3+1} - 1 = 15$

Minimum # of nodes



Maximum # of nodes



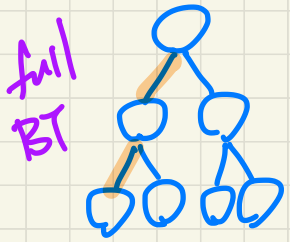
BT Properties: Bounding Height of Tree

Given a *binary tree* with *n* **nodes**, the ***height*** *h* is bounded as:

$$\log_2(n+1) - 1 \leq h \leq n - 1$$

For example, say n = 7

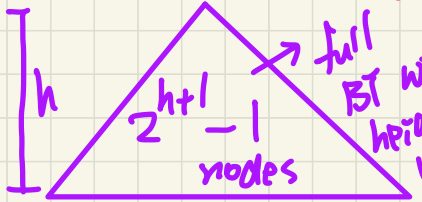
Minimum height



nodes = 7

h = 2.

$\log_2(7+1) - 1 = 2$



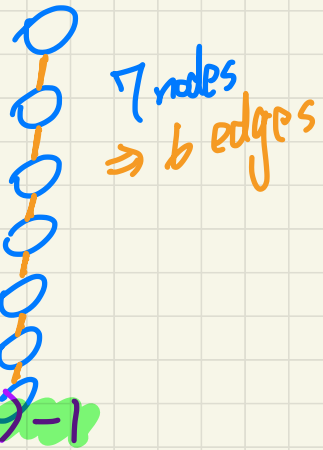
$n = 2^{h+1} - 1$

$n+1 = 2^{h+1}$

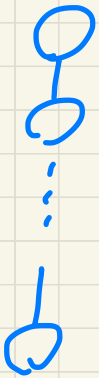
$\log_2(n+1) = h+1$

$h = \log_2(n+1) - 1$

Maximum height



7 nodes ⇒ 6 edges



n nodes ↳ (n−1) edges

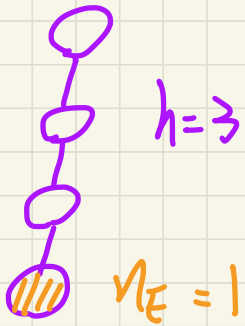
BT Properties: Bounding # of External Nodes

Given a *binary tree* with ***height*** h, the ***number of external nodes*** n_E is bounded as:

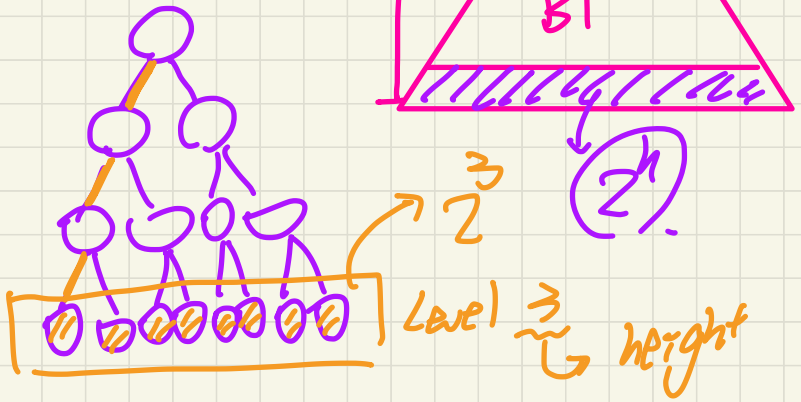
$$1 \leq n_E \leq 2^h$$

For example, say h = 3

Minimum # of External Nodes



Maximum # of External Nodes



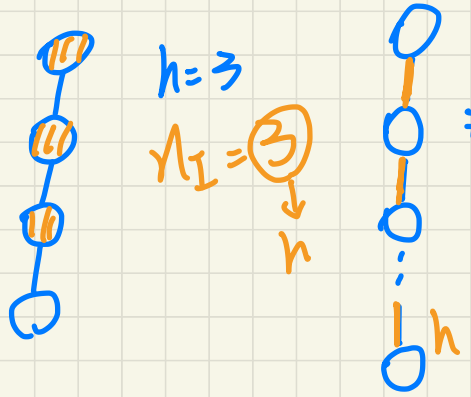
BT Properties: Bounding # of Internal Nodes

Given a *binary tree* with ***height*** h, the ***number of internal nodes*** n_I is bounded as:

$$h \leq n_I \leq 2^h - 1$$

For example, say h = 3

Minimum # of Internal Nodes



Maximum # of Internal Nodes