Data Structures and Algorithms

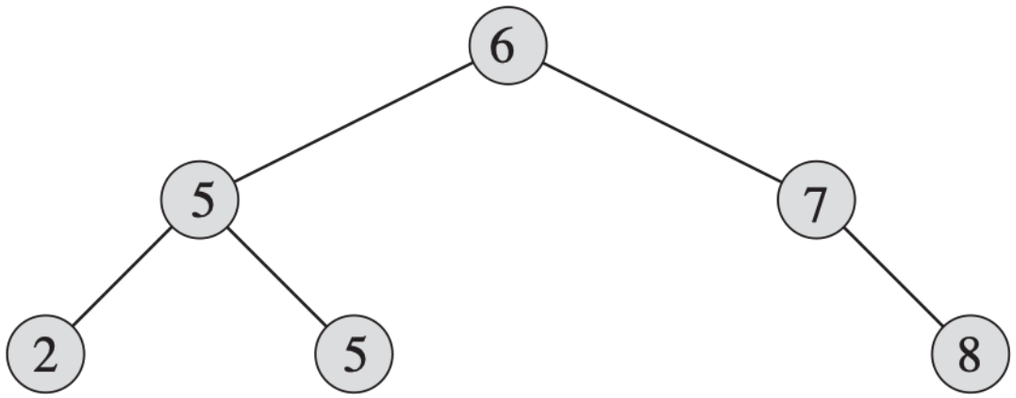
Lecture 20: Binary Trees

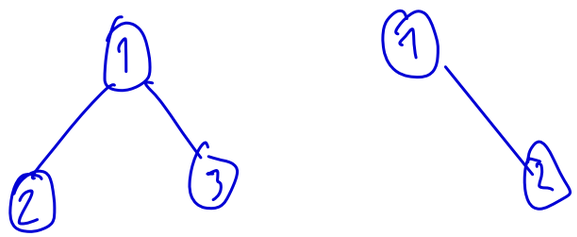
Nopadon Juneam
Department of Computer Science
Kasetsart university

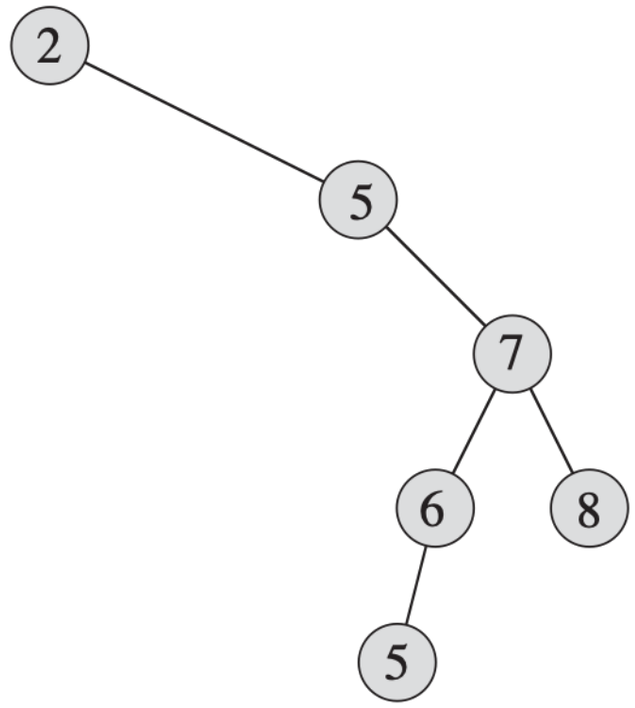
Outlines

- Binary trees: basic terminology and notations
- Properties of binary trees
- Data structures for representing binary trees
  - Linked structure
  - Array-based structure

Binary Trees



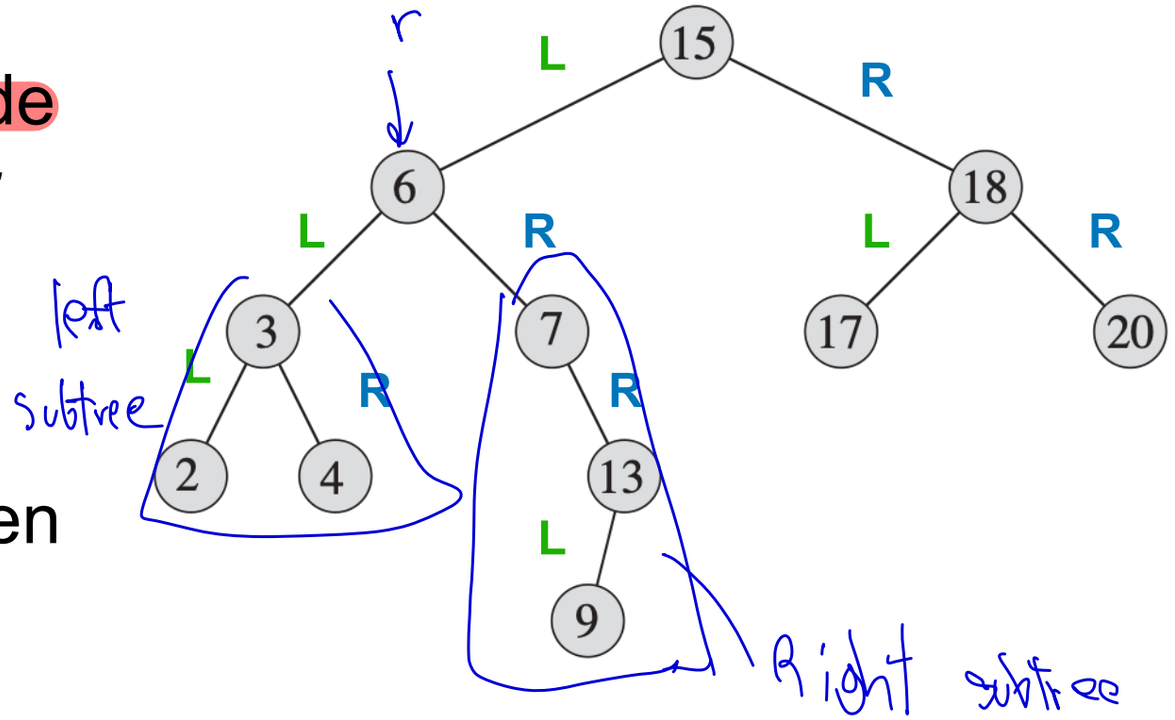




- A ***binary tree*** is kind of an ordered tree in which every node has at most two children. However, if a node has just one child, the position of the child matters

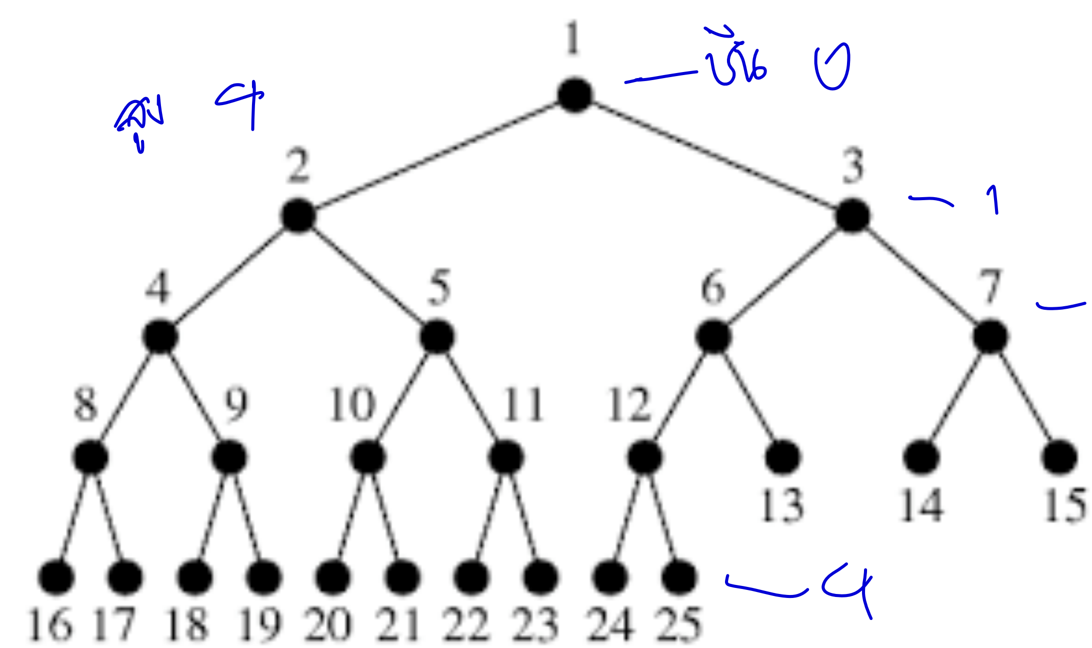
Binary Tree Terminology (1)

- In a binary tree, every child node is labeled as being either a ***left child*** or a ***right child***
  - A left child *precedes* a right child in the ordering of children of a node
- The subtree rooted at a left or right child of an internal node is called the node's ***left subtree*** or ***right subtree***, respectively

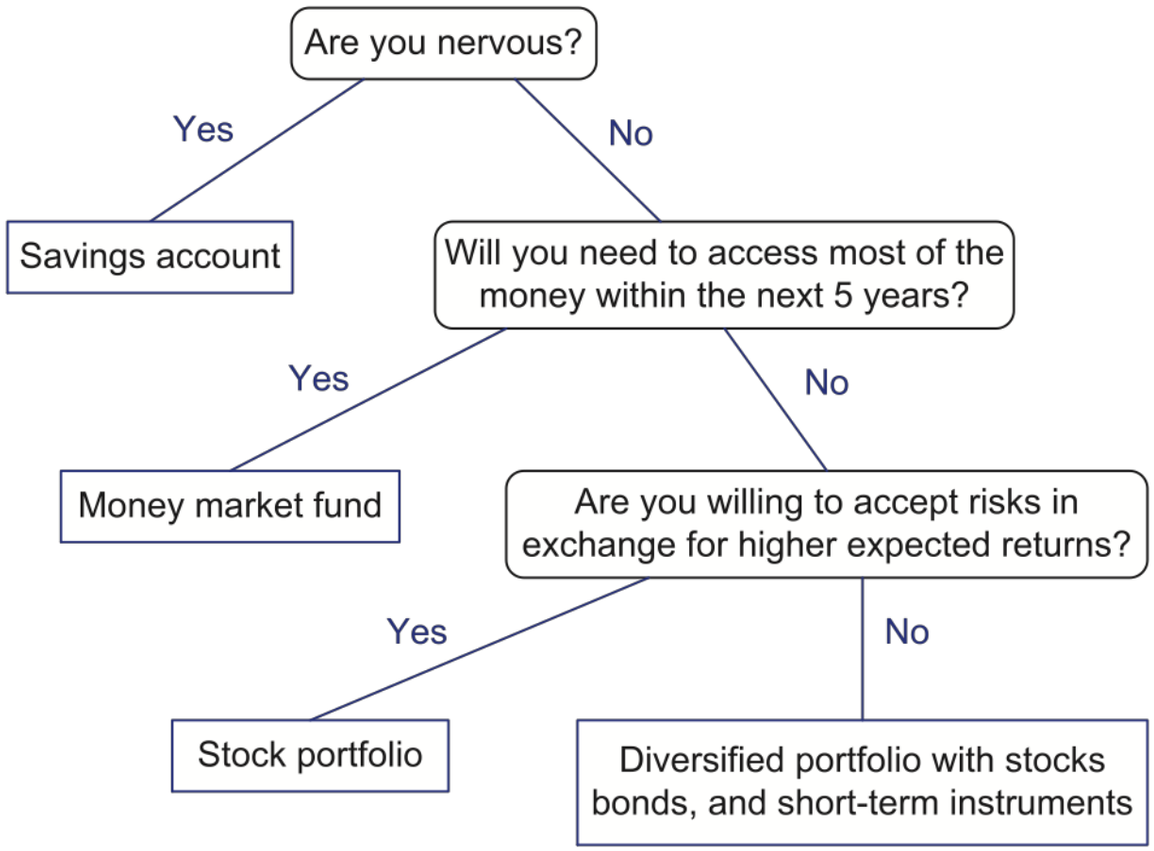


Binary Tree Terminology (2)

- A binary tree is ***proper*** if each node has either zero or two children
  - Some people also refer to such trees as being ***full*** binary trees
  - Thus, in a proper binary tree, every internal node has exactly two children. A binary tree that is not proper is ***improper***



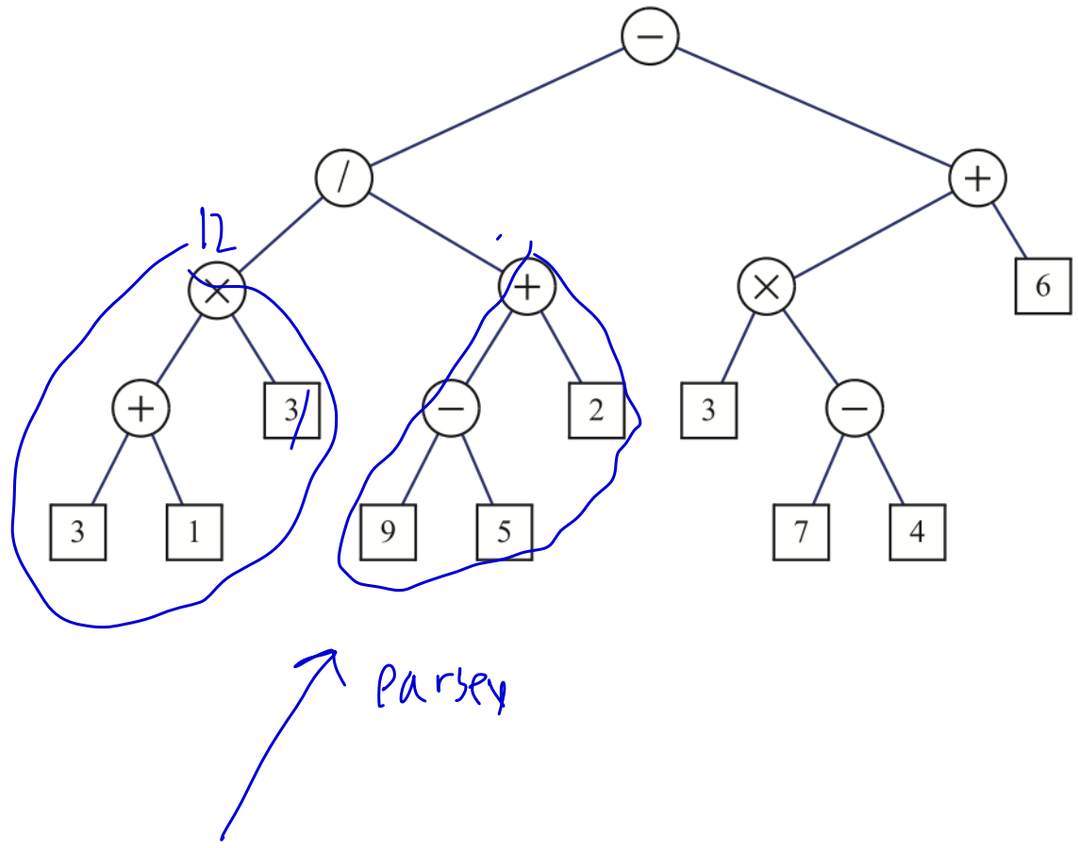
Decision Trees: Class of Binary Trees



- An important class of binary trees called **decision trees** arises in contexts where we wish to represent several different outcomes that can result from answering a series of yes-or-no questions

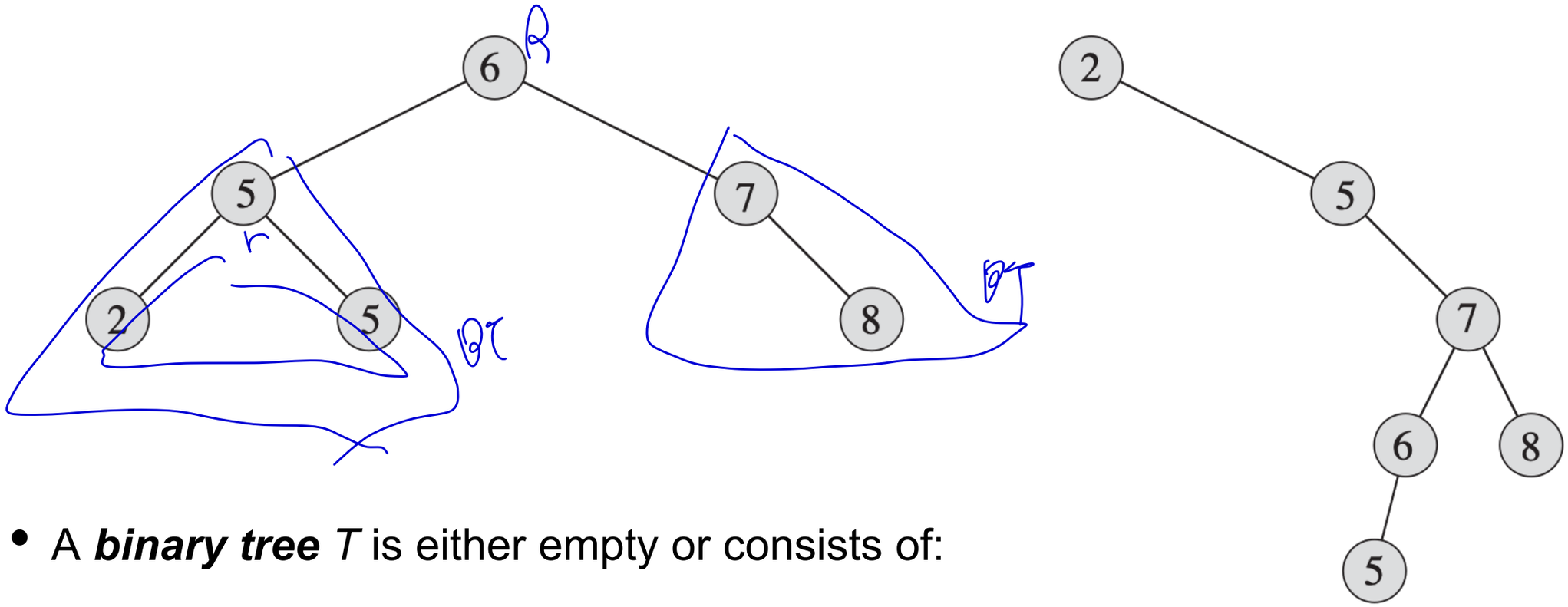
Some Application of Binary Trees

- A binary tree can be used to represent an *arithmetic expression:*
  - Each node in such a tree has a value associated with it
  - If a node is external, then its value is defined by the value of variable or constant
    - If a node is internal, then its value is defined by the result obtained from applying the operation at the node with the operands being the values at its children
- The above tree represents the expression ((((3+1)×3)/((9−5)+2))−((3×(7−4))+6))
- The value associated with the internal node labeled “-” is -13

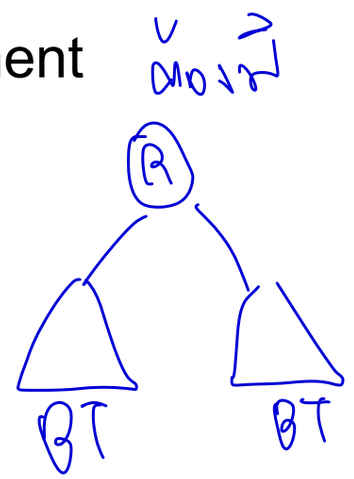


smallest BT = Empty tree

Binary Trees (Recursive Definition)

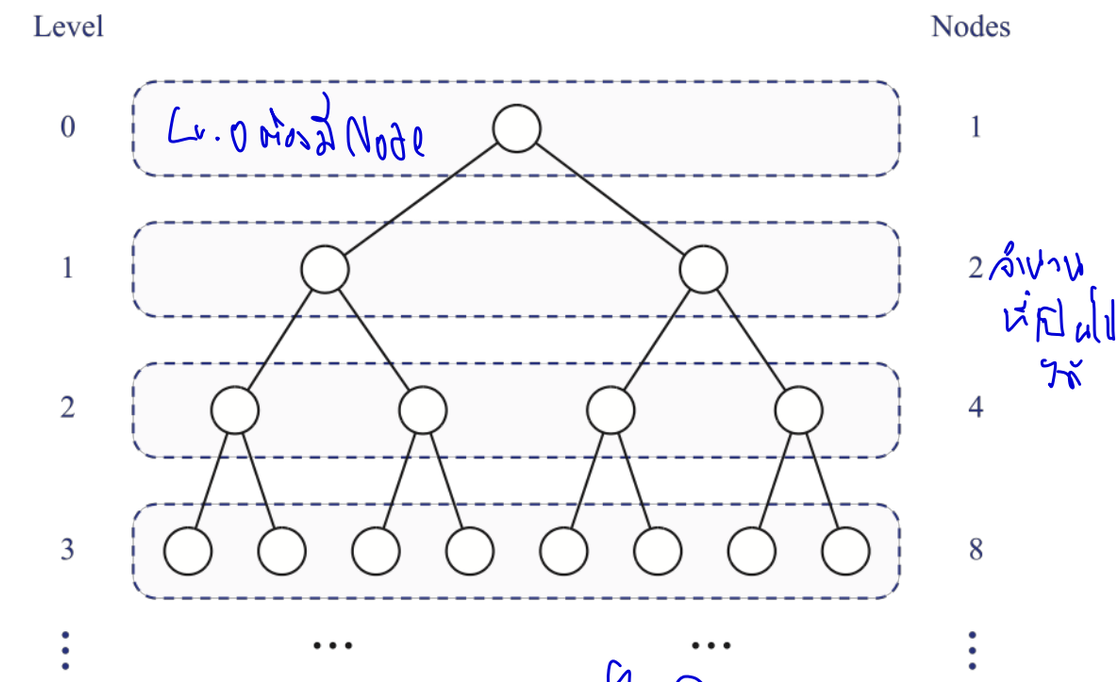


- A ***binary tree*** *T* is either empty or consists of:
  - A node r, called the ***root*** of *T* and storing an element
  - A binary tree, called the ***left subtree*** of *T*
  - A binary tree, called the ***right subtree*** of *T*



Binary Trees's Properties (1)

- A binary tree has interesting properties regarding the *heights* and the *number of nodes* of the tree.
  - We call the set of all nodes of a binary tree T, at the same depth *d*, as the ***level*** *d* of *T:*
    - Level 0 has one node (the root)
    - Level 1 has at most two nodes (the children of the root)
    - Level 2 has at most four nodes, and so on
    - In general, level d has at most 2^d nodes
- *Remark*: The maximum number of nodes on the levels of a binary tree grows exponentially as we go down the tree



Claim: $\sum_{d=0}^{h} 2^d = 2^{h+1} - 1$

Proof: ใช้หลักการ Induction

Base case: $h = 0$

LHS → $\sum_{d=0}^{0} 2^d = 2^0 = 1$

RHS → $2^{0+1} - 1 = 1$

} Base case เป็นจริง

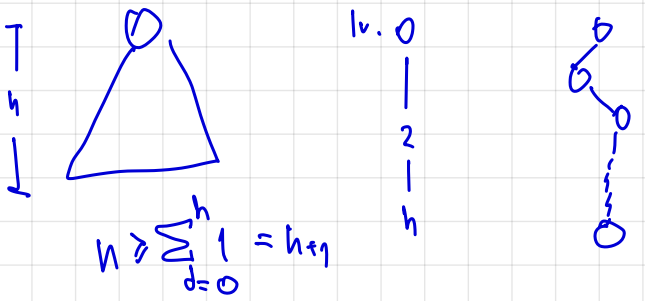
Inductive step:

- Inductive hypothesis:

สมมติว่า $\sum_{d=0}^{h} 2^d = 2^{h+1} - 1$ เป็นจริง

เราจะแสดงว่า $\sum_{d=0}^{h+1} d = 2^{(h+1)+1} - 1$ เป็นจริง

$$\sum_{d=0}^{h+1} 2^d = \underbrace{2^0 + 2^1 + \cdots + 2^h}_{2^{h+1}-1} + 2^{h+1} = 2^{h+1} - 1 + 2^{h+1} = 2^{h+2} - 1 = 2^{(h+1)+1} - 1$$

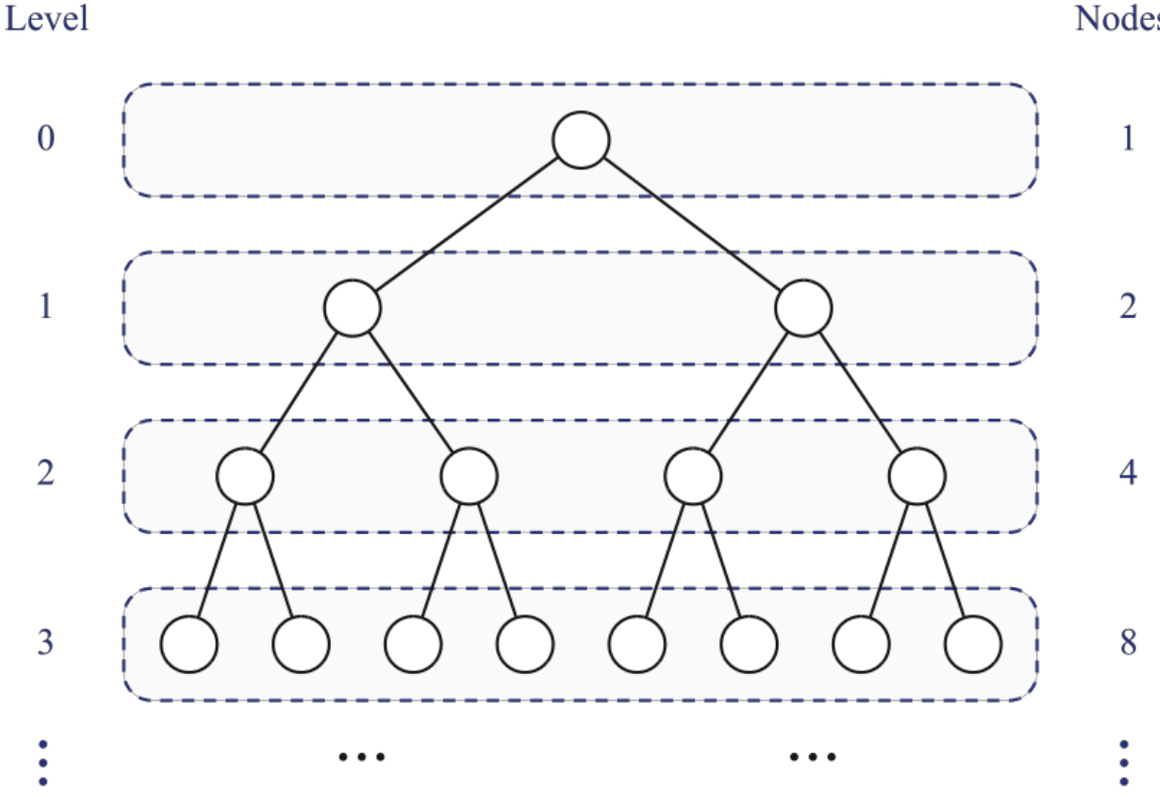


$h \geq \sum_{d=0}^{h} 1 = h+1$

Binary Trees's Properties (2)

- **Proposition 1**: Let T be a nonempty binary tree. Let n, n_E , n_I and h denote the number of nodes, number of external nodes, number of internal nodes, and height of T, respectively. Then, T has the following properties:

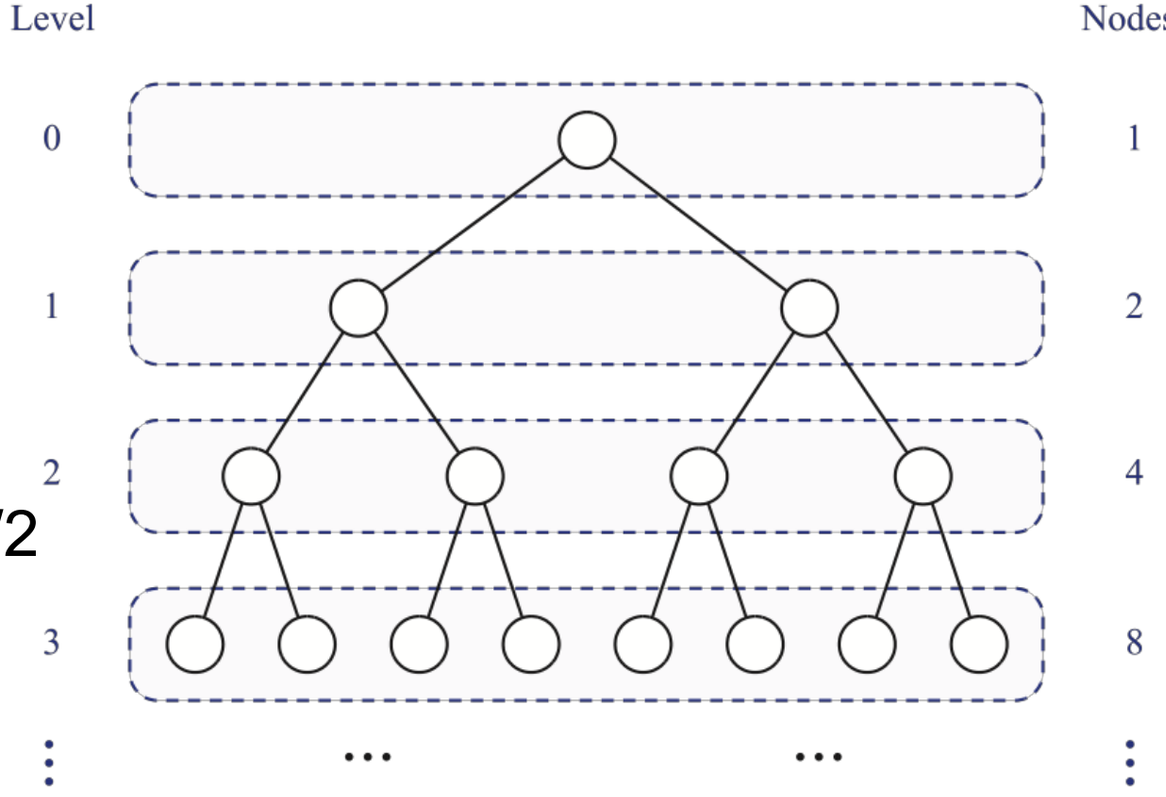
1. $h+1 \le n \le 2^{h+1}-1$
2. $1 \le n_E \le 2^h$
3. $h \le n_I \le 2^h - 1$
4. $\log_2(n+1)-1 \le h \le n-1$



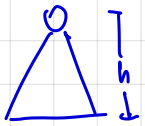
Binary Trees's Properties (3)

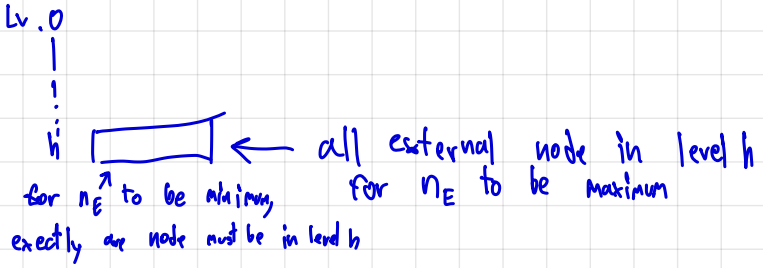
- **Proposition 2**: Let T be a *proper* binary tree. Let n, n_E , n_I and h denote the number of nodes, number of external nodes, number of internal nodes, and height of T, respectively. Then, T has the following properties:

1. $2h+1 \leq n \leq 2^{h+1}-1$
2. $h+1 \leq n_E \leq 2^h$
3. $h \leq n_I \leq 2^h - 1$
4. $\log_2(n+1)-1 \leq h \leq (n-1)/2$



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





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

$$\begin{array}{r} h+1 \leq n \leq 2^{h+1}-1 \\ 1 \leq n_E \leq 2^h \\ \hline h \leq n_I \leq 2^h - 1 \end{array}$$

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

$h+1 \leq n$ | $n \leq 2^{h+1} - 1$
$h \leq n-1$ | $n+1 \leq 2^{h+1}$
$\log_2(n+1) \leq h+1$
$\log_2(n+1) - 1 \leq h$

Linked Structure for Binary Trees

- In a linked structure for a binary tree *T*, we represent each node of *T* by an object *p* with the following fields:
  - A reference to the node's element
  - A link to the node's parent
  - A link to the node's two children