

# Mathematical Properties of Trees with Applications to Lower Bound Theory

Textbook Reading:

*Algorithms: Foundations and Design Strategies*

Chapter 4, Section 4.2, pp. 143-149

Appendix, A.7, pp. 460-463.

*Algorithms: Special Topics*

Chapter 6. Section 6.2,

pp. 169-181



# Mathematical Properties of Trees

- Mathematical properties of trees are important in the design and analysis of algorithms.
- In this lecture we obtain
  - a result relating the number of nodes and number of edges of a general tree
  - lower bound for the depth of a binary tree in terms of the number of nodes
  - relation between the number of leaf nodes and the total number of nodes in a 2-tree
  - lower bound for the depth of a binary tree in terms of the number of leaf nodes
  - Lower bound for leaf-path length
  - Application to obtain a lower bound for the worst-case and average complexities of the problem of comparison-based sorting

# Number of edges vs. number of nodes in a tree

**Theorem.** The number  $m$  of edges of any tree  $T$  is one less than the number  $n$  of nodes.

**Proof by Induction.** We perform induction on the number of  $n$  of nodes.

**Basis Step.** A tree with one node has no edges, i.e., we have  $m = 0 = n - 1$ .

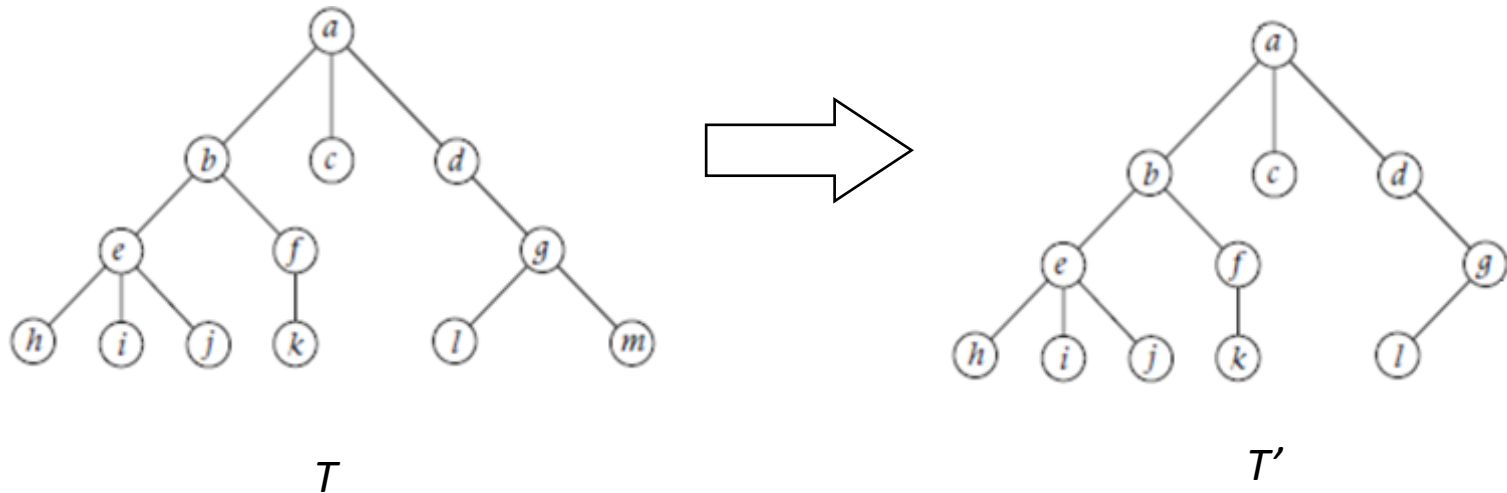
# Induction Step

Assume true for  $n = k$ , i.e., any tree having  $k$  nodes has  $k - 1$  edges.

Now consider a tree  $T$  having  $n = k + 1$  nodes. For convenience, let  $n(T)$  and  $m(T)$  denote the number of nodes and edges of  $T$ .

Note that any node at the deepest level (highest level number) must necessarily be a leaf. Thus,  $T$  contains at least one leaf node.

Remove a leaf node (and its incident edge) to obtain a tree  $T'$  having  $k$  nodes

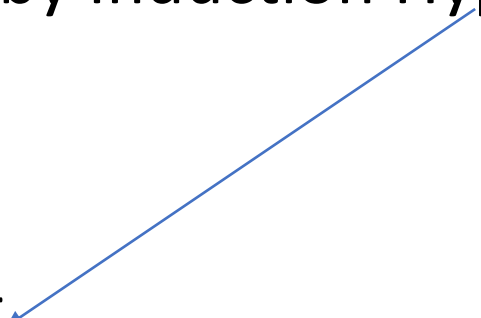


# Applying Induction Hypothesis

Since  $T'$  has  $k$  nodes, we can apply the induction hypothesis (inductive assumption)

$$m(T') = n(T') - 1 \quad (\text{by Induction Hypothesis})$$

Thus, we have

$$\begin{aligned} m(T) &= m(T') + 1 \\ &= (n(T') - 1) + 1 \quad (\text{substituting}) \\ &= n(T') = n(T) - 1. \end{aligned}$$


This completes the induction step and the proof.

## Lower Bound on Depth of a Binary Tree

**Proposition 4.2.2** Every binary tree with  $n$  nodes has depth  $d$  at least

$$\lceil \log_2 n \rceil$$

Further, equality is achieved for the complete binary tree.

# Proof

Let  $n_i$  be the number of nodes at level  $i$ ,  $i = 0, 1, \dots, d$ .

Since tree is binary, each node has at most 2 children, so that

$$n_i \leq 2n_{i-1}, i = 1, 2, \dots, d \text{ and } n_0 = 1.$$

An easy induction proves that

$$n_i \leq 2^i, i = 0, 1, \dots, d. \quad (1)$$

Clearly,

$$n = n_0 + n_1 + \dots + n_d. \quad (2)$$

Substituting (1) into (2) and using result  $x^0 + x^1 + \dots + x^d = (x^{d+1} - 1)/(x - 1)$ :

$$n \leq 2^0 + 2^1 + \dots + 2^d = 2^{d+1} - 1.$$

We have

$$2^{d+1} \geq n + 1$$

$$\Rightarrow d + 1 \geq \log_2(n + 1)$$

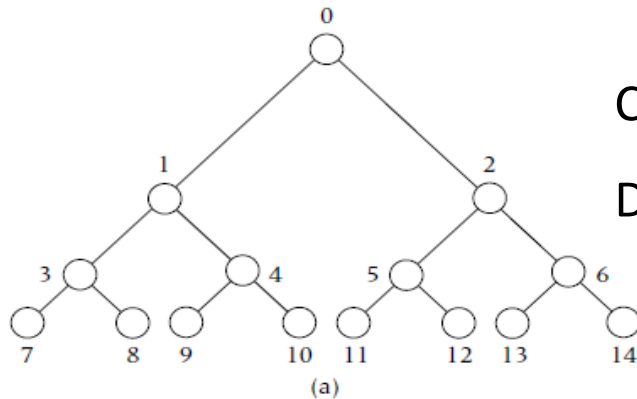
$$\Rightarrow d \geq \log_2(n + 1) - 1.$$

Using fact that  $d$  is an integer it is easy to show that  $d \geq \lfloor \log_2 n \rfloor$ .



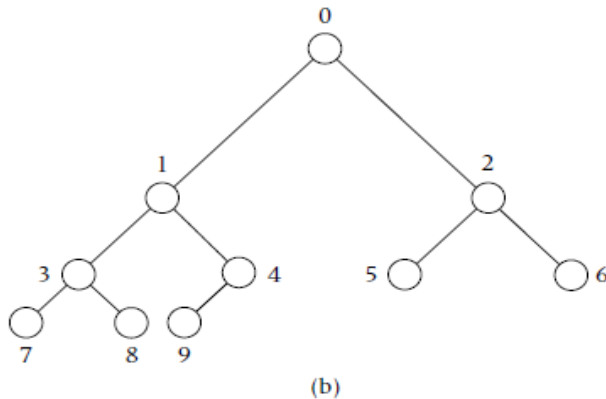
# Complete Tree is an Extremal Tree

The minimum is achieved by the complete tree on  $n$  nodes.



Complete tree on  $n = 15$  nodes is a full tree.

Depth is  $\lfloor \log_2 n \rfloor = \lfloor \log_2 15 \rfloor = 3$

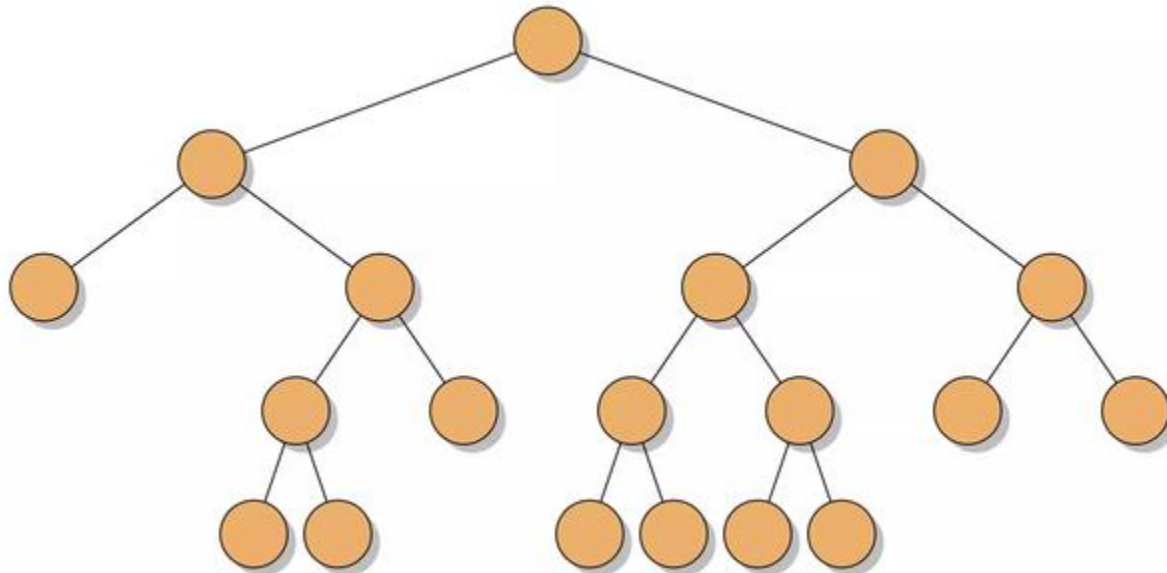


Complete tree on  $n = 10$  nodes.

Depth is  $\lfloor \log_2 n \rfloor = \lfloor \log_2 10 \rfloor = 3$

# 2-tree

A **2-tree** is a tree where every node that is not a leaf has exactly two children.



An **internal** node is a node that is not a leaf node.

Let  $I(T)$  and  $L(T)$  denote the number of internal and leaf nodes of a 2-tree  $T$ , respectively. For convenience, let  $I = I(T)$  and  $L = L(T)$ .

**Proposition 4.2.3** Let  $T$  be a 2-tree. Then,  $L = I + 1$ .

Clearly, the total number  $n$  of nodes satisfies  $n = I + L$ , so we have:

**Corollary 1.**  $n = 2I + 1$ .

**Corollary 2.**  $n = 2L - 1$ .

Parametrizing the induction. We must decide, which parameter, we will perform induction on, i.e., the number  $I$  of internal nodes, the number  $L$  of leaf nodes or the total number  $n$  of nodes. We will choose  $L$ .

# Basis Step

The proposition is true for  $L = 1$ . A single node 2-tree has one leaf node, the root, and 0 internal nodes, so we have

$$L = 1 = I + 1.$$

# Induction Step

Assume proposition is true for  $L = k$ , i.e., all 2-trees  $T$  with  $k$  leaf nodes have  $k - 1$  internal nodes.

Now consider **any** 2-tree  $T$  having  $k + 1$  leaf nodes.


PSN. To apply the induction hypothesis, we need to perform an operation that reduces  $T$  to a tree  $T'$  with  $k$  leaf nodes.

How to do this?

Since  $T'$  has one fewer leaf nodes than  $T$ , i.e.,  $T'$  has  $k$  leaf nodes, we can apply the induction hypothesis, i.e.,

$$L(T') = I(T') + 1.$$

Thus,


$$L(T) = L(T') + 1 = (I(T') + 1) + 1 = I(T) + 1$$

This completes the induction step and the proof of the Proposition.

# Lower bound on depth of a binary tree in terms of number of leaf nodes

**Proposition 4.2.6** Every binary tree with  $L$  leaf nodes has depth  $d$  at least

$$\lceil \log_2 L \rceil$$

Further, equality is achieved for the complete binary tree.



# Proof

First assume the binary tree  $T$  is a 2-tree. By Proposition 4.2.2

$$d \geq \lfloor \log_2 n \rfloor \quad (1)$$

And by Corollary 2 of Proposition 4.2.3

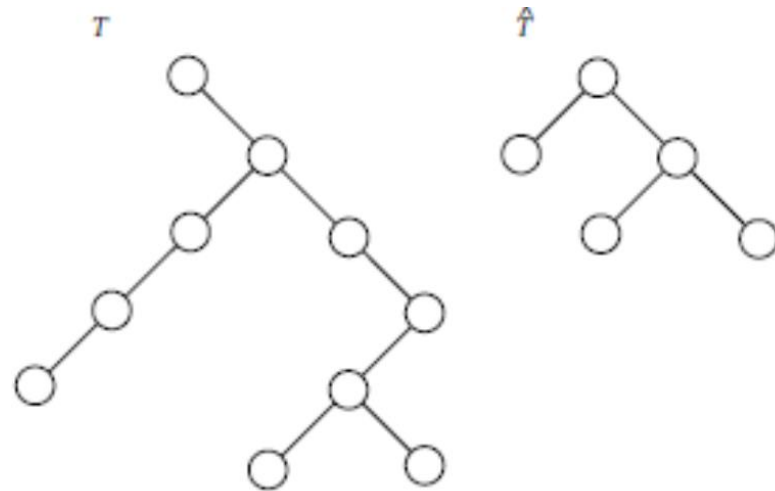
$$n = 2L - 1 \quad (2)$$

Substituting (1) into (2) we obtain

$$d \geq \lfloor \log_2 n \rfloor = \lfloor \log_2 (2L - 1) \rfloor = \lfloor \log_2 L \rfloor$$

# Transformation yield result for general binary tree

Now consider a general binary tree  $T$ . Construct  $\hat{T}$  from  $T$  as follows



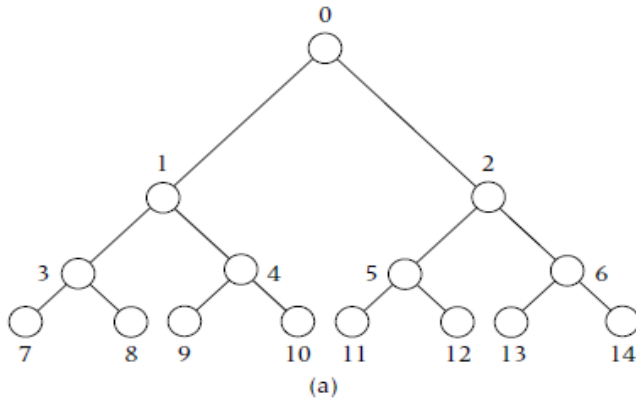
By argument on previous slide  $d(\hat{T}) \geq \lfloor \log_2 L(\hat{T}) \rfloor$

Thus,

$$d(T) \geq d(\hat{T}) \geq \lfloor \log_2 L(\hat{T}) \rfloor = \lfloor \log_2 L(T) \rfloor$$

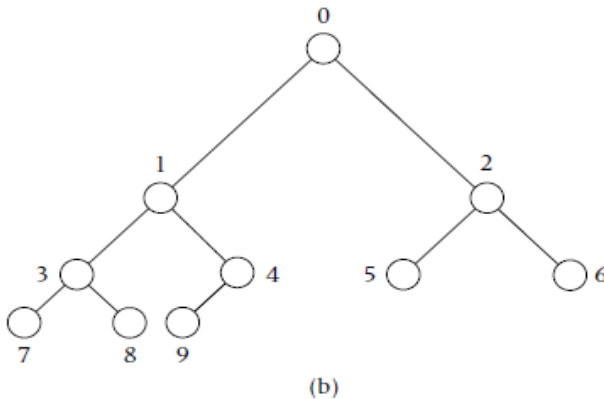
# Complete tree is an Extremal Tree

The minimum is achieved by the complete tree



Complete tree on  $n = 15$  nodes has  $L = 8$  leaf nodes.

Depth is  $\log_2 L = \log_2 8 = 3$

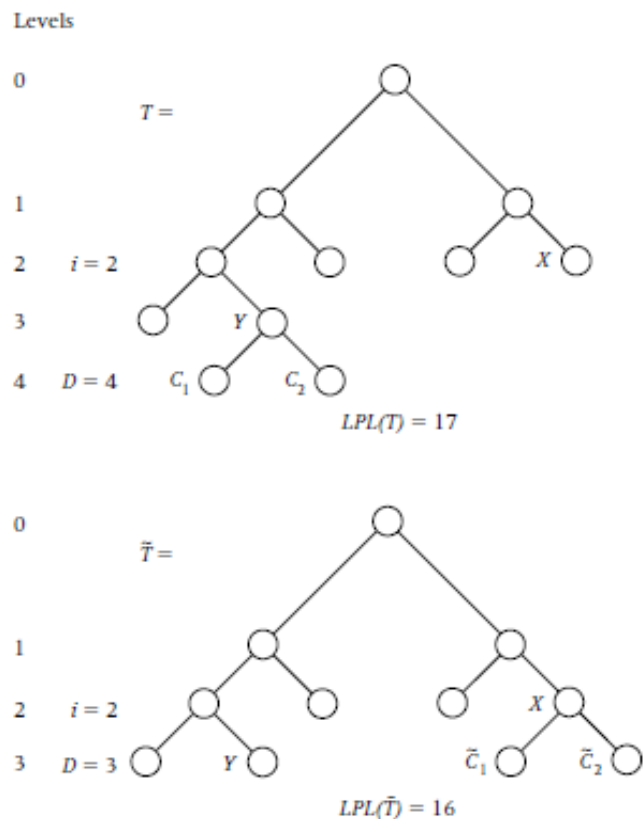


Complete tree on  $n = 10$  nodes has  $L = 5$  leaf nodes.

Depth is  $\lceil \log_2 L \rceil = \lceil \log_2 5 \rceil = 3$

# Leaf-Path Length

The leaf-path length of a binary tree  $T$ , denoted by  $LPL(T)$  is the sum of the lengths over all paths from the root to a leaf node.



# Lower bound on Leaf-Path Length

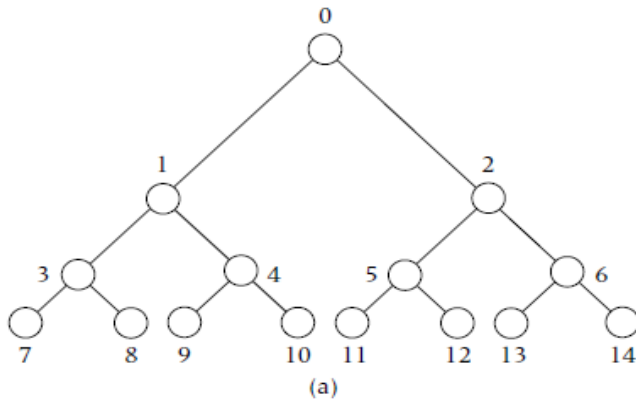
**Theorem.** For any binary tree  $T$ ,

$$LPL(T) \geq \lceil L \log_2 L \rceil .$$

Further, equality is achieved for a full binary tree.

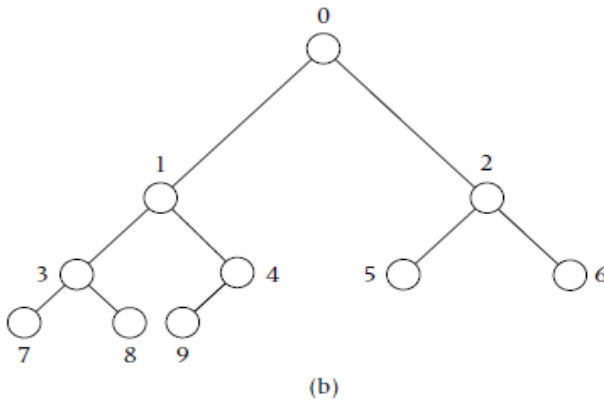
# Complete tree is an extremal tree

The minimum is achieved for the complete tree



Complete tree on  $n = 15$  nodes has  $L = 8$  leaf nodes.

Depth is  $\log_2 L = \log_2 8 = 3$



Complete tree on  $n = 10$  nodes has  $L = 5$  leaf nodes.

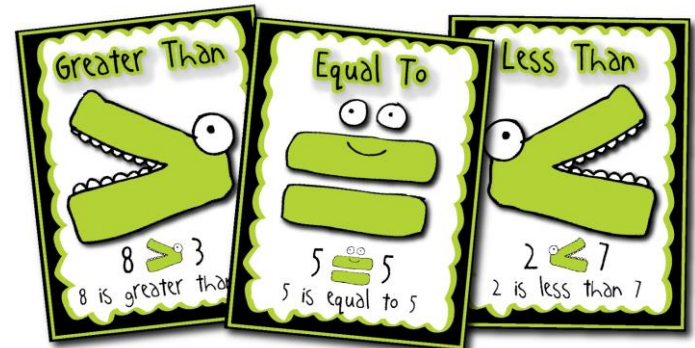
Depth is  $\lceil \log_2 L \rceil = \lceil \log_2 5 \rceil = 3$

# Application to Lower Bound Theory

- Given a problem and a specific algorithm for solving the problem, it is important to determine how close the algorithm comes to exhibiting optimal worst-case or average complexity for the problem itself.
- But how do we determine what optimal behavior is for the problem?
- The complexity of any algorithm that correctly solves the problem gives us upper bounds for the complexities of the problem. The object in lower bound theory is to determine good lower bound estimates for the worst-case and average complexities of problems.
- The results we obtained for binary trees is a useful tool in obtaining these lower bounds.

# Decision and Comparison Trees

- **Decision trees** are an important and commonly used method of establishing lower bounds.
- A **comparison tree** is a decision tree modeling a comparison-based algorithm an internal node in corresponds to a comparison made by the algorithm using one of the comparison operators  $<$ ,  $=$ ,  $>$ .
- External leaf nodes correspond to outputs of the algorithm. We assume that any nodes in the comparison tree that are never reached by any input to the algorithm are pruned.

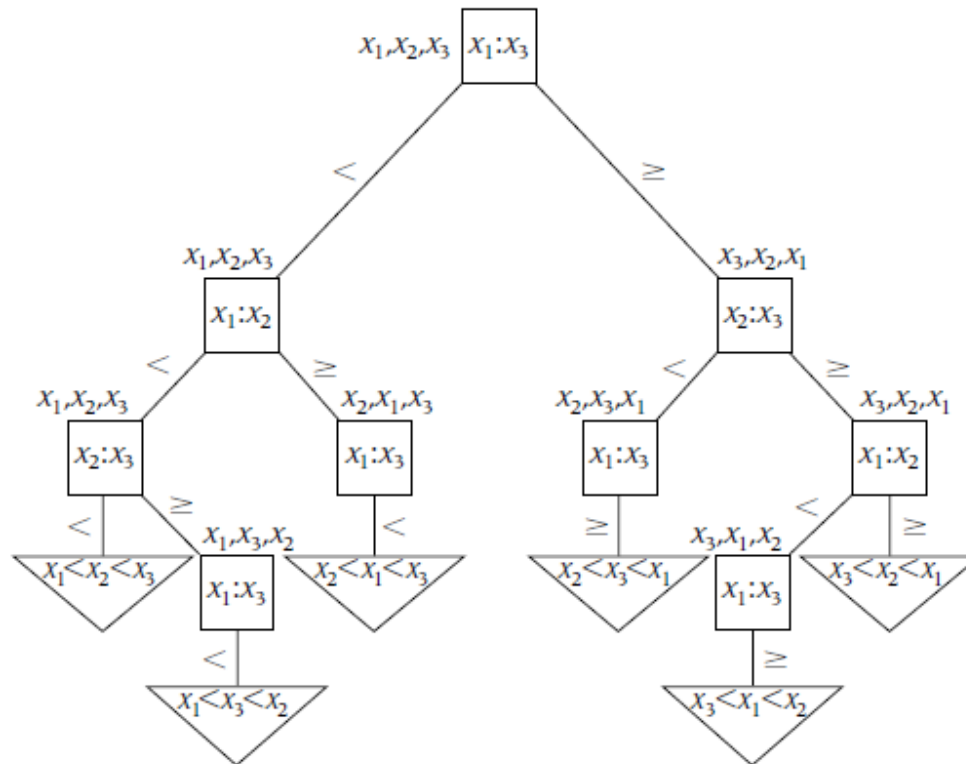




# Comparison-Based Sorting Algorithms

## Key Observation:

A comparison tree for any given comparison-based algorithm for inputs of size  $n$ , will have  $n!$  leaf nodes.



# Worst-Case Complexity Lower Bound

## Second Key Observation:

The worst-case complexity  $W(n)$  equals the depth  $D(T)$  of the comparison tree  $T$ , i.e.,

$$W(n) = D(T).$$

By our result for binary trees we have that

$$D(T) \geq \lceil \log_2 L(T) \rceil \geq \log_2 L(T) = \log_2 n! \in \Omega(n \log n)$$

Thus,

$$W(n) \in \Omega(n \log n)$$

# Average Complexity Lower Bound

## Third Key Observation:

The average complexity  $A(n)$  equals the leaf-path length  $LPL(T)$  divided by  $n!$ , i.e.,

$$A(n) = LPL(T)/n!$$

By our result for binary trees we have that

$$LPL(T) \geq L \log_2 L(T) \geq n! \log_2 n!$$

Thus,

$$A(n) = LPL(T)/n! = \log_2 n! \in \Omega(n \log n)$$

**Why did the pine tree get in trouble?**

Because it was being knotty.

