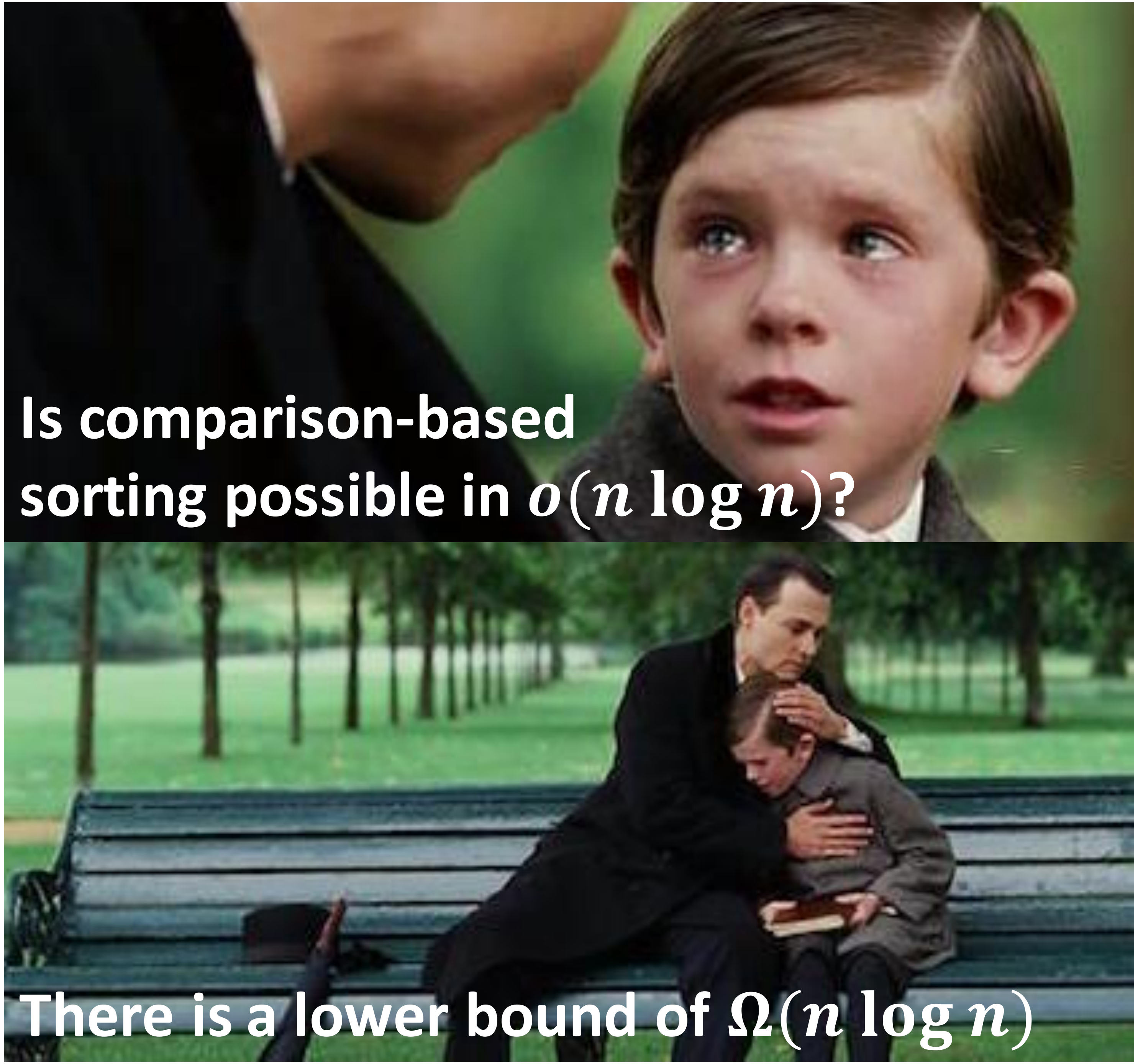
# Algorithms & Data Structures I
# CSC 225

Ali Mashreghi

Fall 2018

Department of Computer Science, University of Victoria


Is comparison-based
sorting possible in $o(n \log n)$?
There is a lower bound of $\Omega(n \log n)$

# Proving a lower-bound

- We want to prove that any comparison sort takes at least $\Omega(n \log n)$ in the worst-case.

- In other words, a comparison sort **cannot guarantee** a running time of $o(n \log n)$ **on all inputs**.

# Comparison-based sorting

- Let’s say we have the following ADT:

**Abstract-Integer:**

***data type:*** An integer $x$

***operations:***

//compares $x$ and $y$, and returns $\leq$ or $>$

compare(Abstract-Integer $y$)

# Comparison-based sorting

- Let's say we have the following ADT:

**Abstract-Integer:**

***data type:*** An integer $x$

***operations:***

//compares $x$ and $y$, and returns $\leq$ or $>$

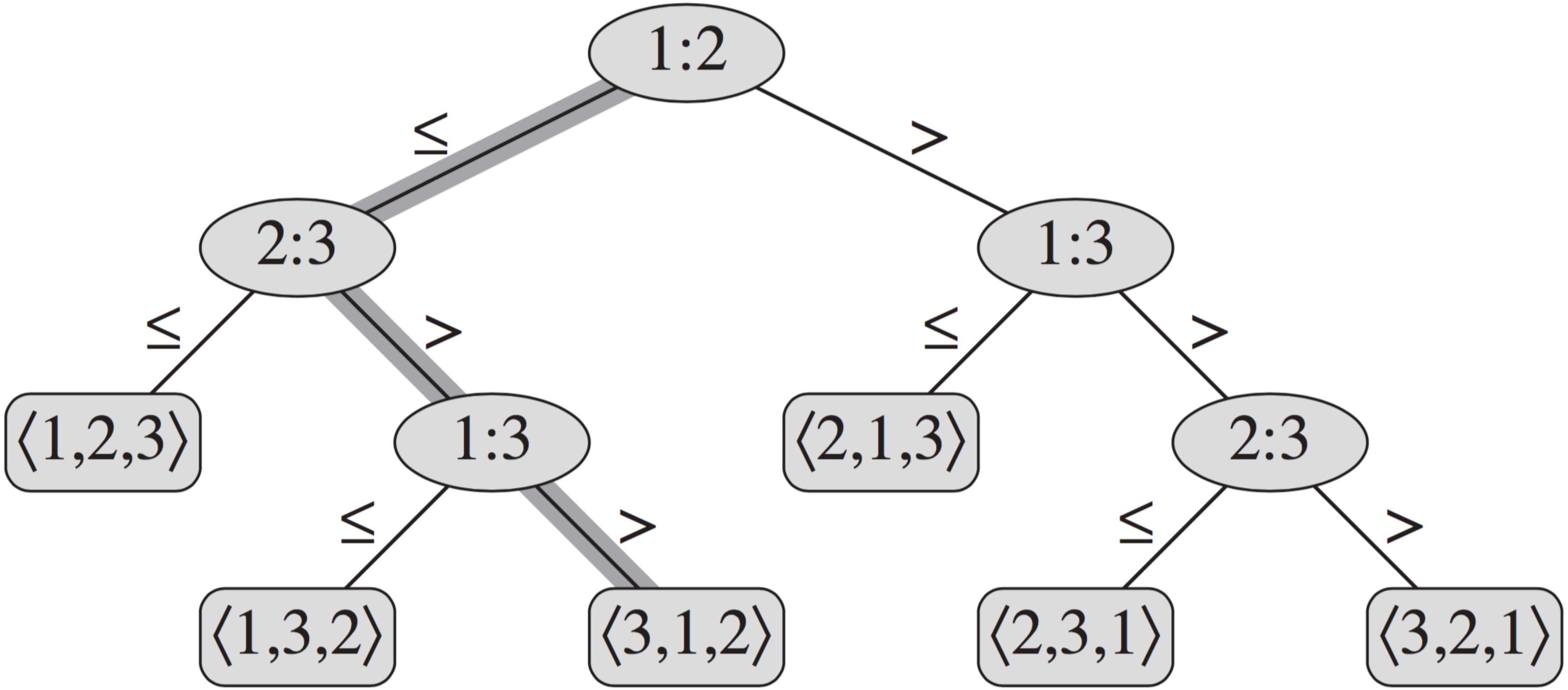
compare(Abstract-Integer $y$)

- We assume a *comparison-based sorting algorithm* (also called a **comparison sort**) is working with instances of this ADT. So, it can't see the actual values.

# Decision Tree Model

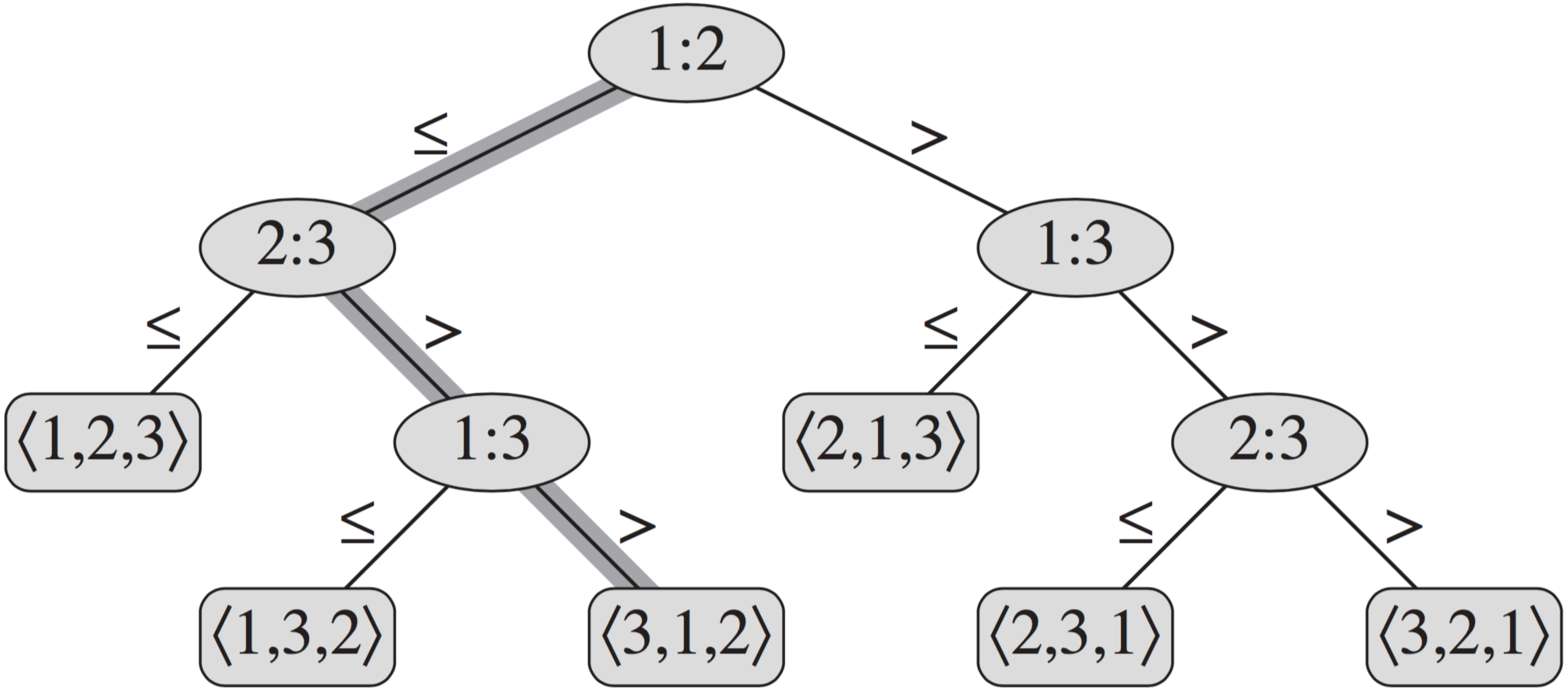
- We can view the behavior of comparison sorts as a decision tree which is a **full binary tree**.
- Each **internal node** is annotated with a comparison.
- And each **leaf node** is annotated with a **permutation** which is the answer to the sorting problem.
- We assume all elements are distinct since we want to get a lower bound for the worst case.
- We assume comparison queries are of the form $A[i] \leq A[j]$.

# Decision Tree Model



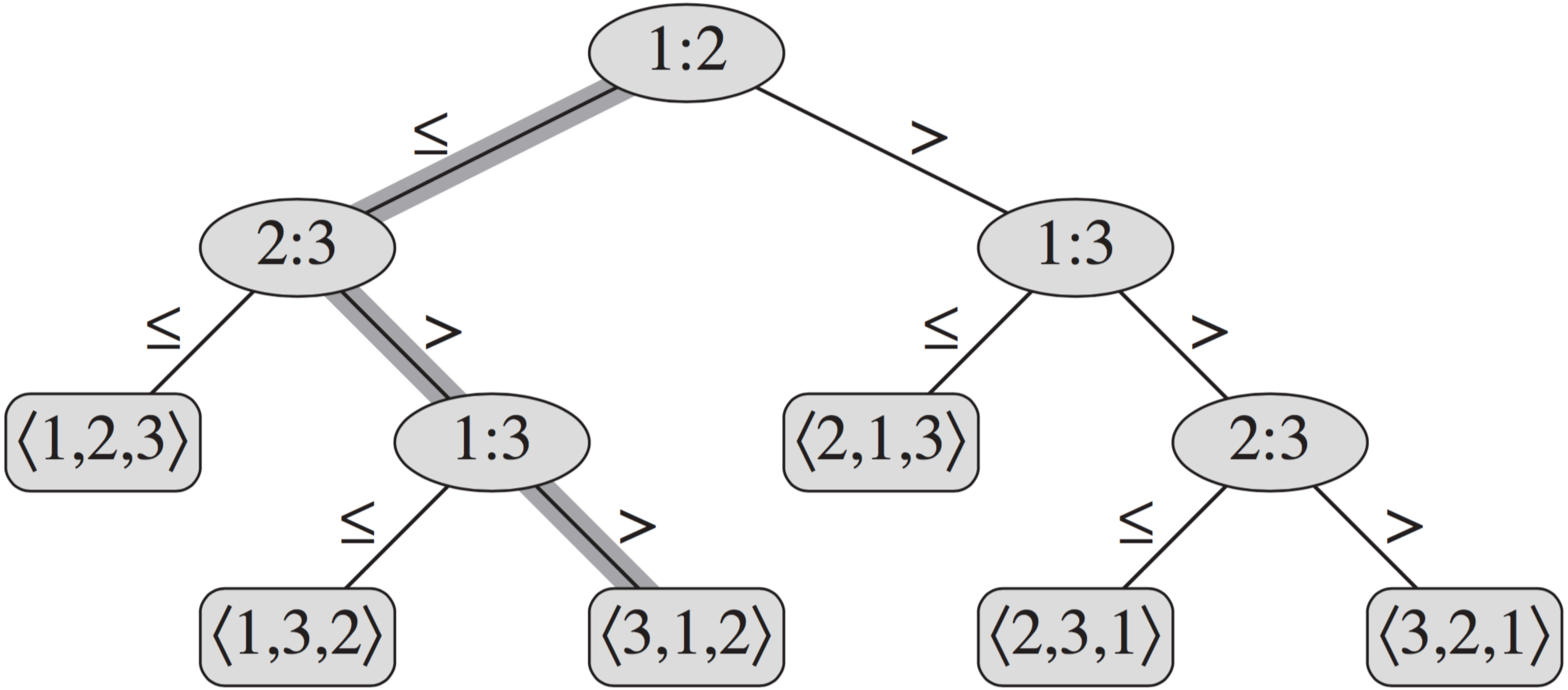

- Example of a decision tree for INSERTION-SORT on $A[1..3]$
- $i$: $j$ means comparing $A[i]$ and $A[j]$

# Decision Tree Model



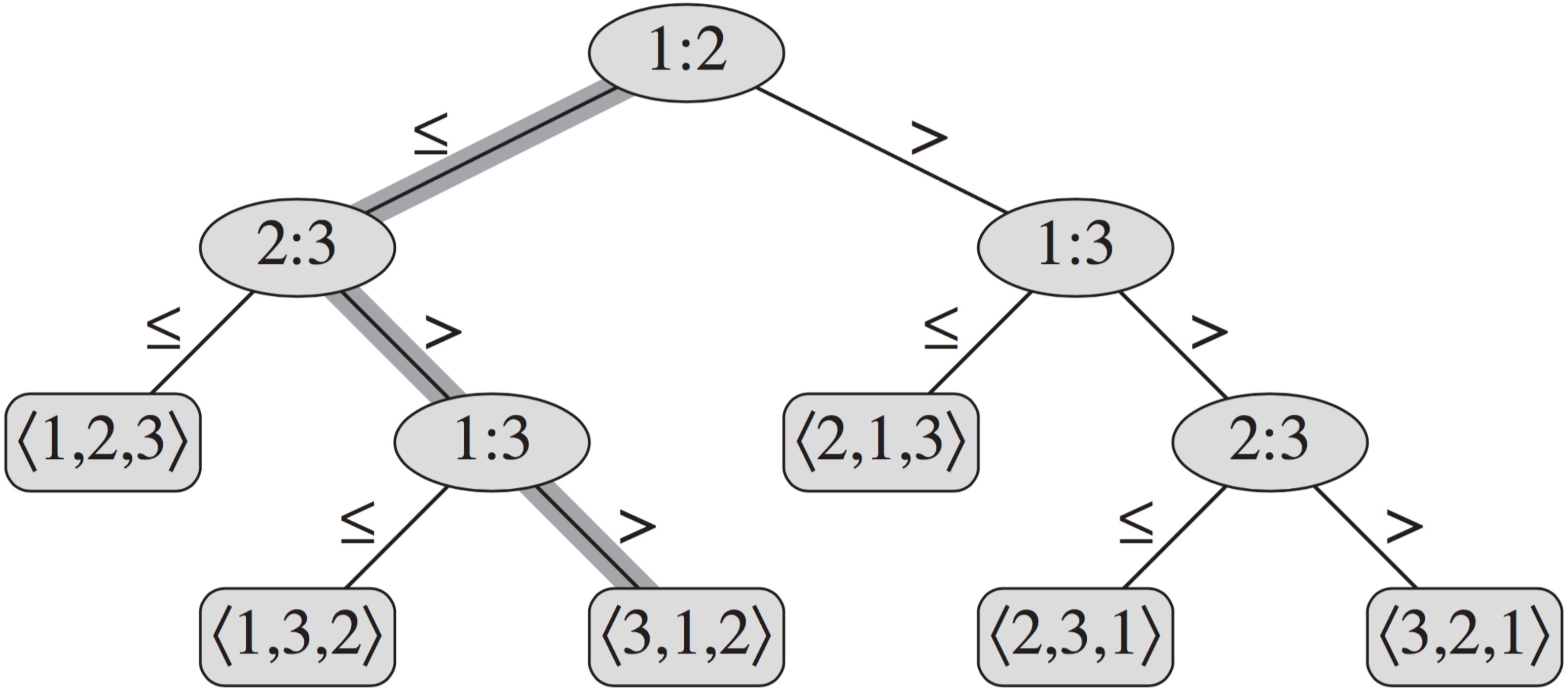

- Each **path** from the **root to a leaf** is the sequence of comparisons required to sort a particular input.
- The input $A = \{7, 11, 5\}$ can cause the **highlighted** path to be followed by the algorithm.

# Decision Tree Model



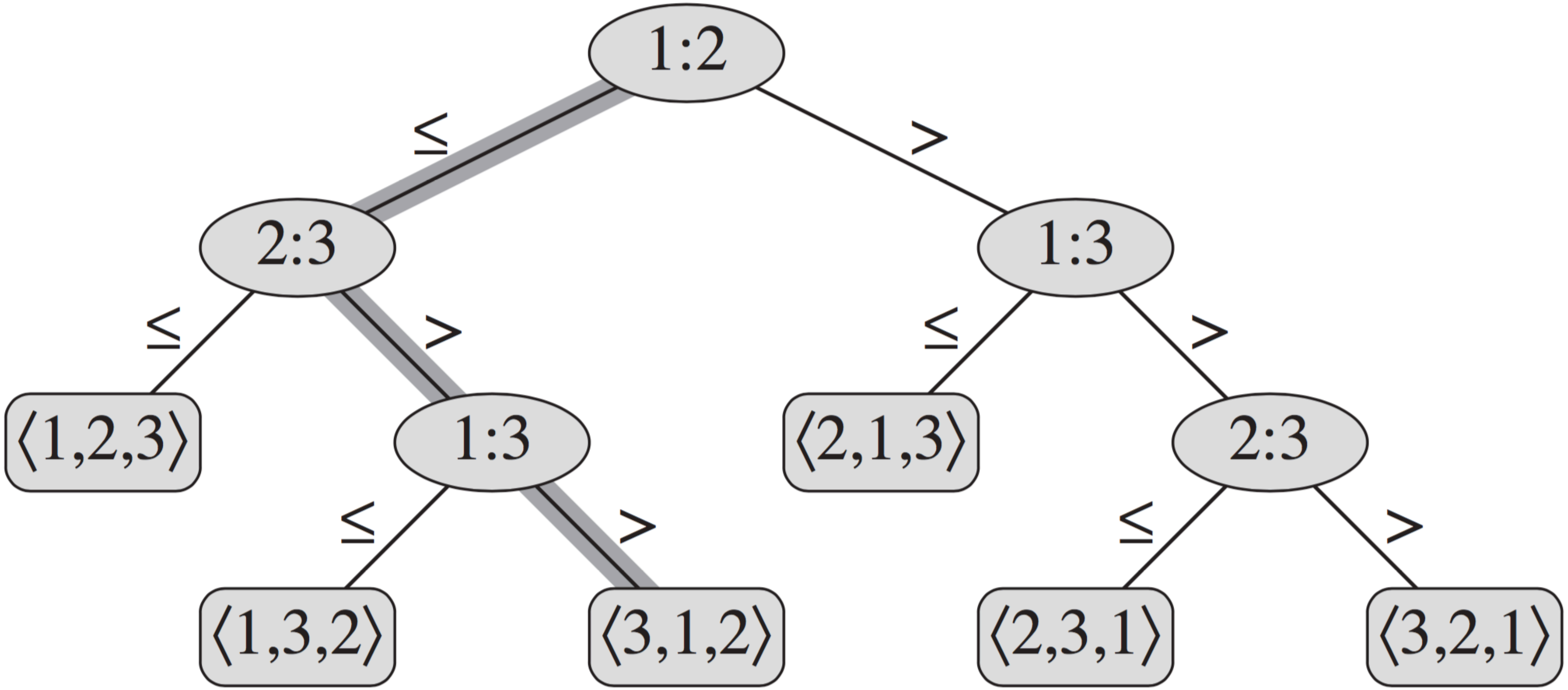

- Each **leaf** is a **re-ordering** of the **original indices** that can make the whole array sorted.

# Decision Tree Model



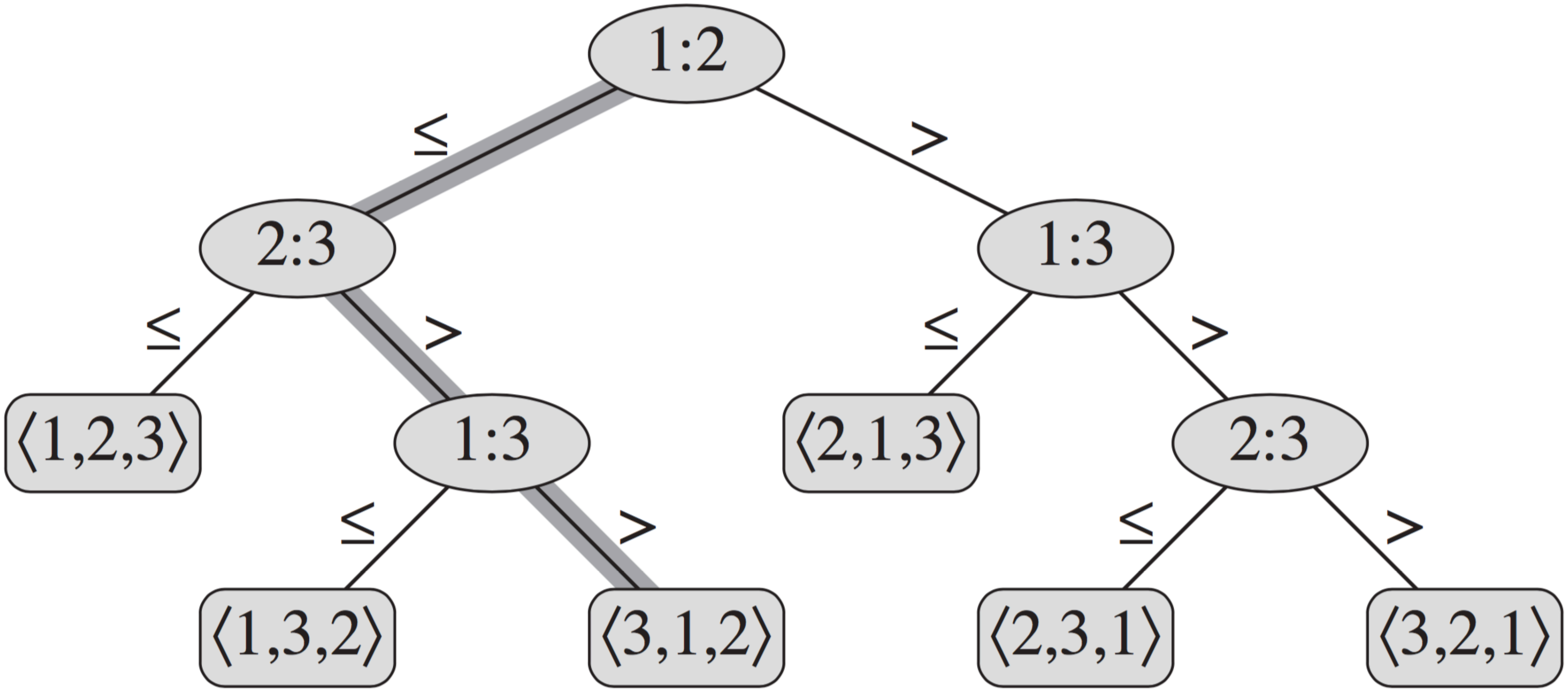

- We use this model to analyze the running time of a sorting algorithm
- **Only comparisons** contribute to the cost (i.e. running time)

# Decision Tree Model



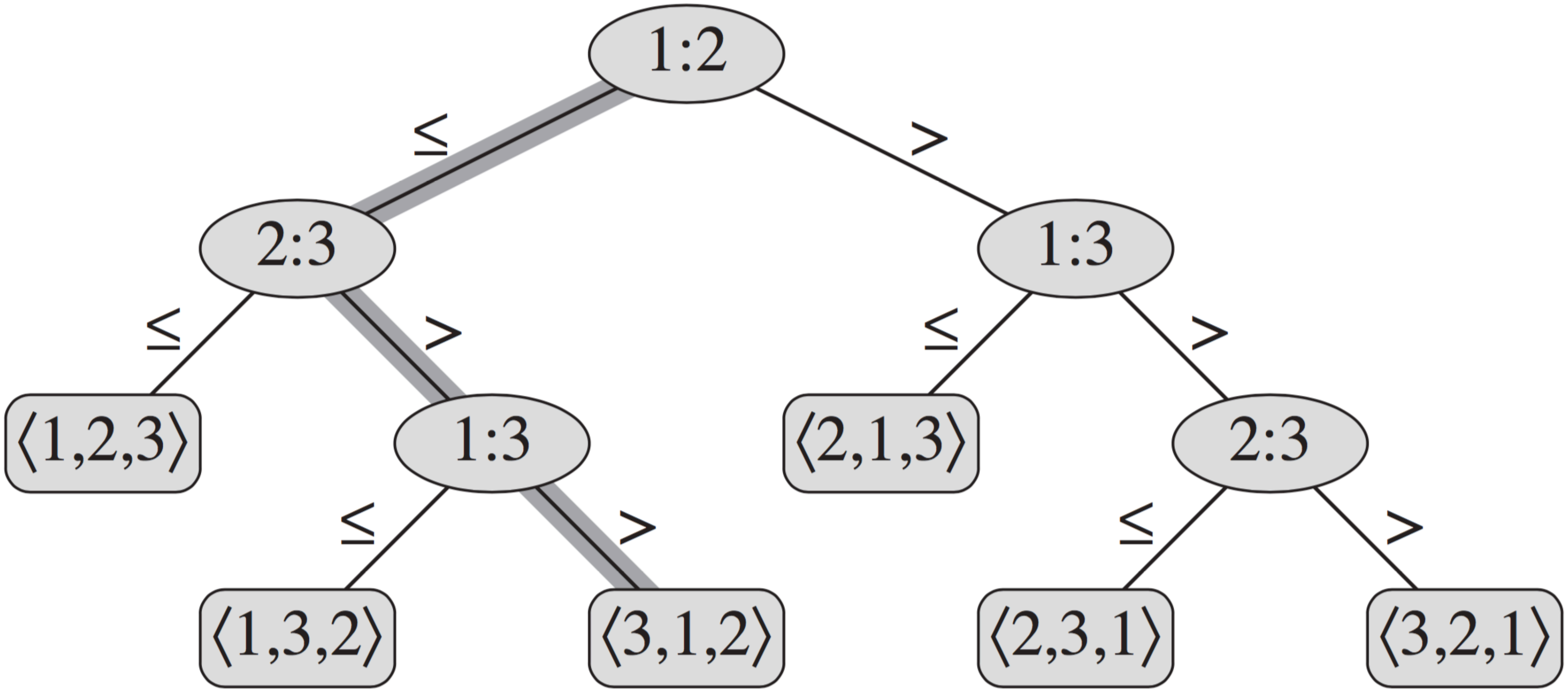

- If we prove that $\boldsymbol{\Omega(n\, log\, n)}$ comparisons are necessary in the worst case input for **any** sorting algorithm, we are done!

# Decision Tree Model



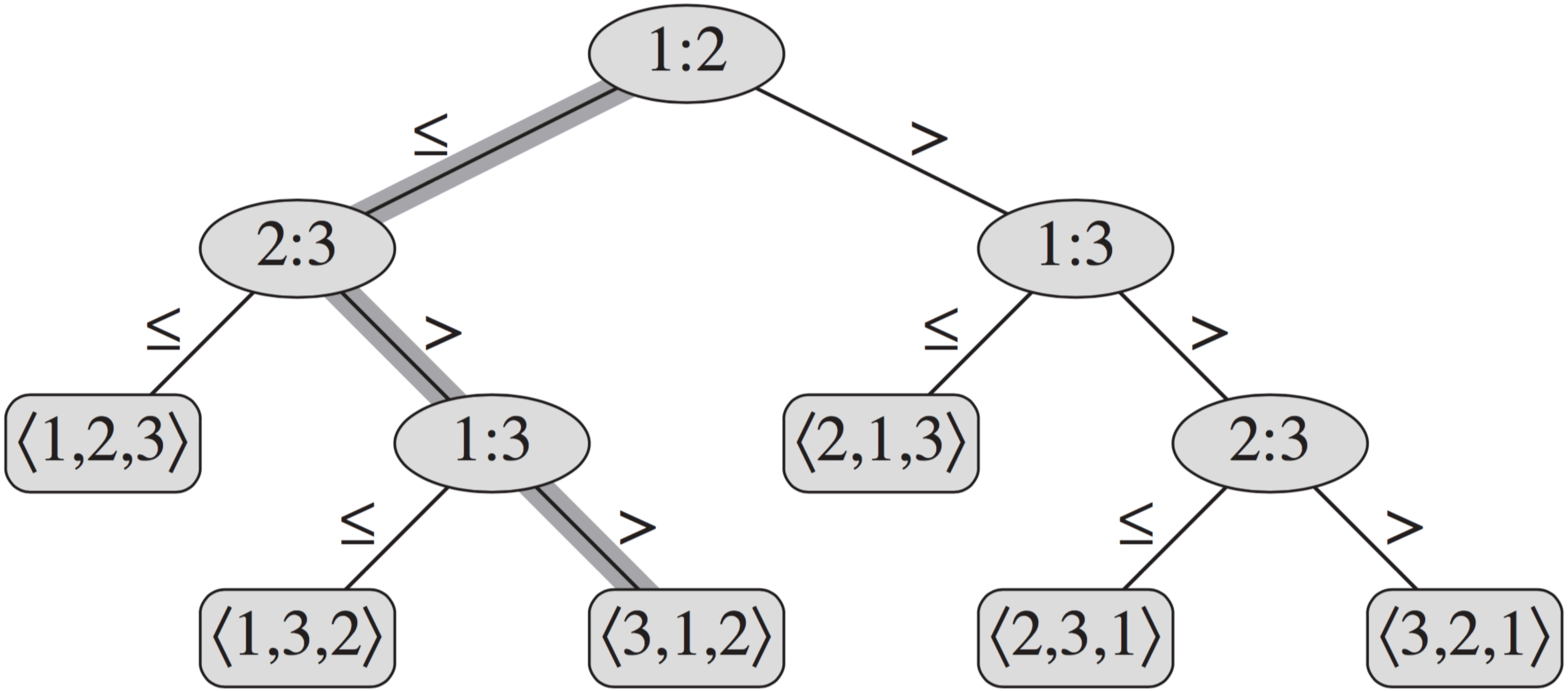

- We have 3 elements and 6 leaves or 6 possible answers.
- **Question:** What’s the connection?

# Decision Tree Model



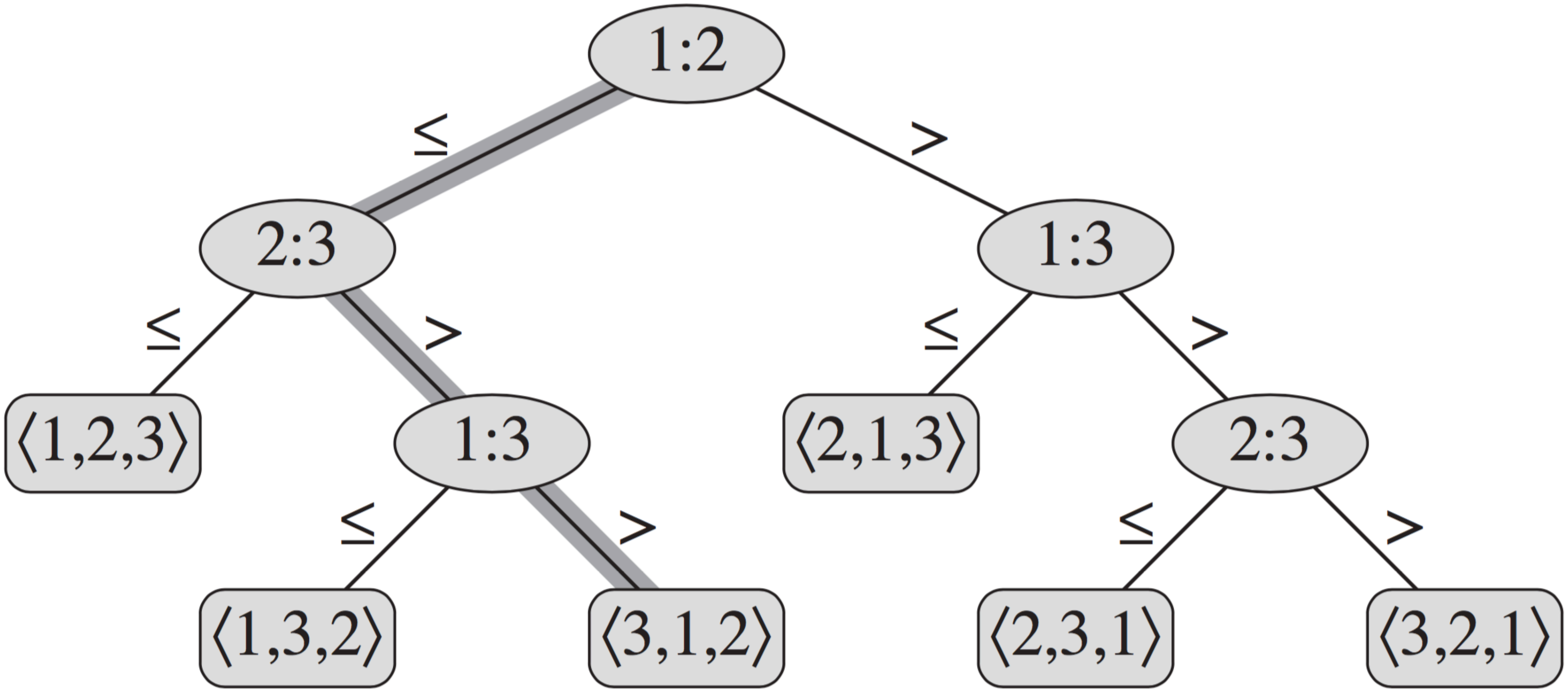

- We have 3 elements and 6 leaves or 6 possible answers.
- **Question:** What's the connection? **Answer:** The tree of a **correct** sorting algorithm **must** have **all permutations** ($\boldsymbol{n}!$) in the leaves.

# Decision Tree Model




- **Question:** What does the **longest root-to-leaf path** correspond to?

# Decision Tree Model




- **Question:** What does the **longest root-to-leaf path** correspond to? **Answer:** It's the **height** of tree which corresponds to the worst-case number of comparisons.

# Proving a lower bound

- The decision tree for a correct sorting algorithm on an input of size $n$, must have $n!$ leaves. (one for each possible permutation)

- A good sorting algorithm tries to keep the height low, so that the worst-case running time is as low as possible.

- **Note:** Proving a lower bound of $\Omega(n \log n)$ here means proving that even the best comparison sort that one can design requires at least $n \log n$ time on some input.

# Proving a lower bound

- **Question:** Knowing that this full binary tree must contain $n!$ leaves, how low could the height be?

# Proving a lower bound

- **Question:** Knowing that this full binary tree must contain $n!$ leaves, how low could the height be?

- **Answer:** We know that in a full binary tree with $k$ leaves, the height is in the range of $\log k$ to $k - 1$. So, the lowest possible height of a tree with $n!$ leaves is $\log n!$ which we prove is $\Omega(n \log n)$. (it's actually $\Theta(n \log n)$, however, since we want to prove a lower bound we use the big-Omega notation.)