Lower Bound for Comparison-based Sorting

How Fast Can We Sort?

- The worst-case running time of the *comparison based* sorting algorithms are

Name of the Algorithm	**Worst Case**
Bubble Sort, Insertion Sort, Selection Sort	$O(n^2)$
Quick Sort	$O(n^2)$
Heap Sort, Merge Sort	$O(n \log n)$

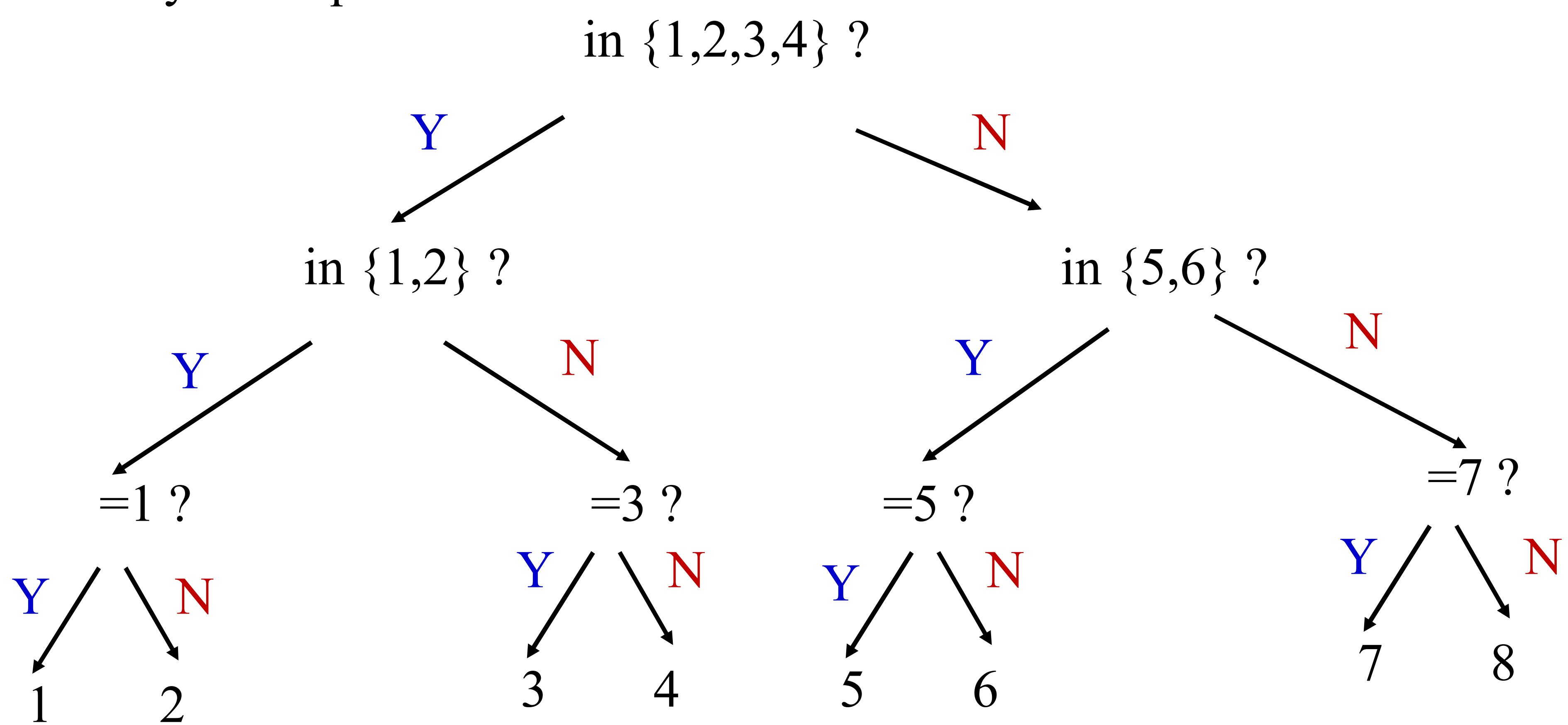
- Can we do better than $O(n \log n)$?
- Can we say that it is impossible to sort faster than $\Omega(n \log n)$ in the worst case?
- If we could, then this would be what it's called a lower bound.

The Lower Bound of an Algorithm

- A lower bound for a problem is the worst-case running time of the best possible algorithm for that problem.
- Lower bounds prove that we cannot hope for a better algorithm, no matter how smart we are.
- So, how to prove a lower bound, without going through all possible algorithms?!?!
- Only very few lower bound proofs are known.

Binary Decision Trees

- Can we <u>always</u> figure out a number from {1, 2, 3, …, 8} by 3 yes/no questions?



- The number of yes/no questions to figure out a number from $\{1, 2, 3, \ldots, n\}$ is $k = \lceil \log_2 n \rceil$

Lower Bound for Comparison-based Sorting

- Sorting by comparisons
  - yes/no questions: is A[i] > A[j] ?

- A decision tree represents the comparisons made by a comparison sort.

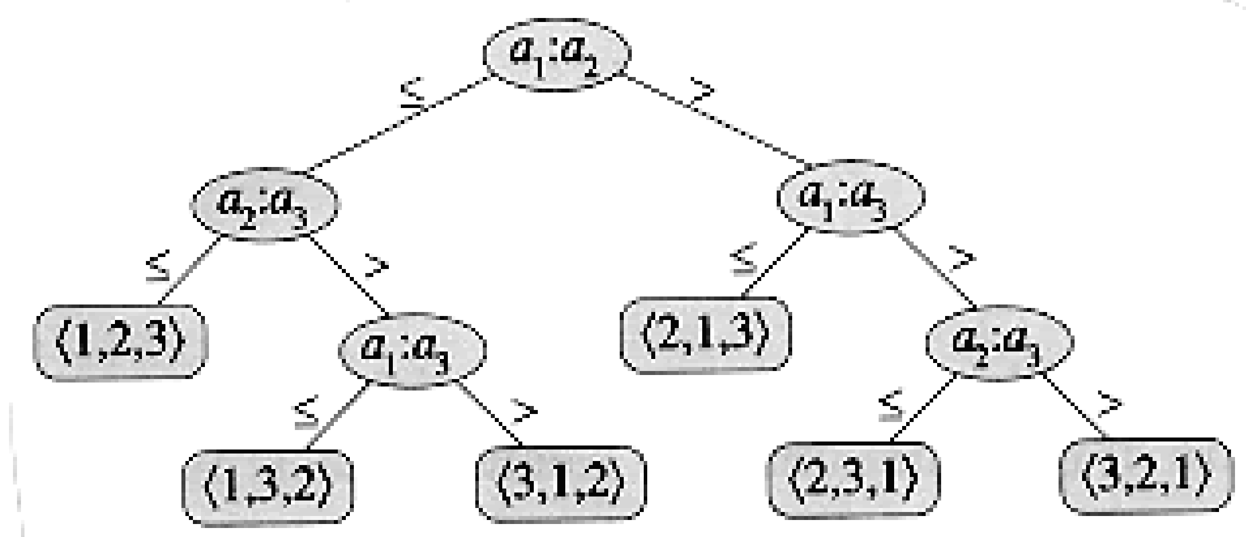
- Theorem:

  Any comparison based sorting algorithm requires $\Omega(n \log n)$ *comparisons in the worst-case.*

  Proof: Using Binary Decision Tree

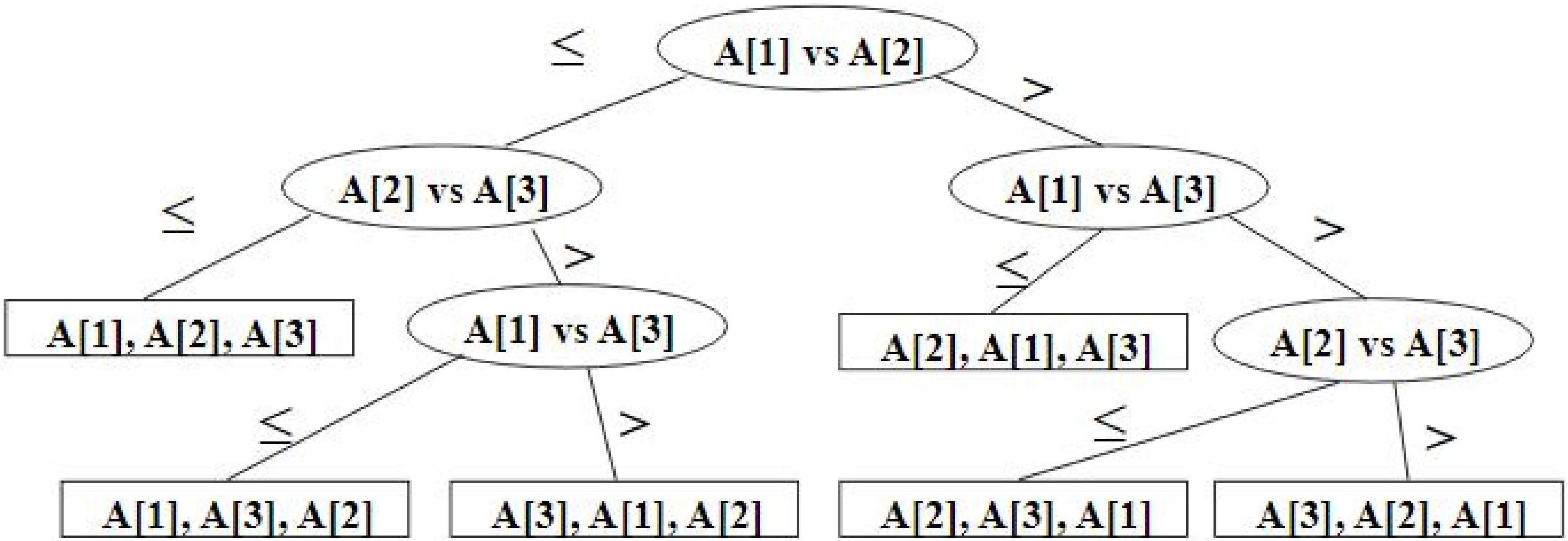
Binary Decision Trees

- Given any comparison-based sorting algorithm, we can represent its behavior on an input of size n by a decision tree.
  - Each internal node in the decision tree corresponds to one of the comparisons in the algorithm.
  - Start at the root and do first comparison:
    - If $\leq$ take left branch, else take right branch, etc.
  - Each leaf represents one possible ordering of the input
    - One leaf for each of n! possible orderings.



Binary Decision Trees

- Example: Sorting $n = 3$ numbers ($3! = 6$ leaves)



- The height of the tree, that is, the length of the longest path from root to a leaf in this tree

  = worst-case number of comparisons

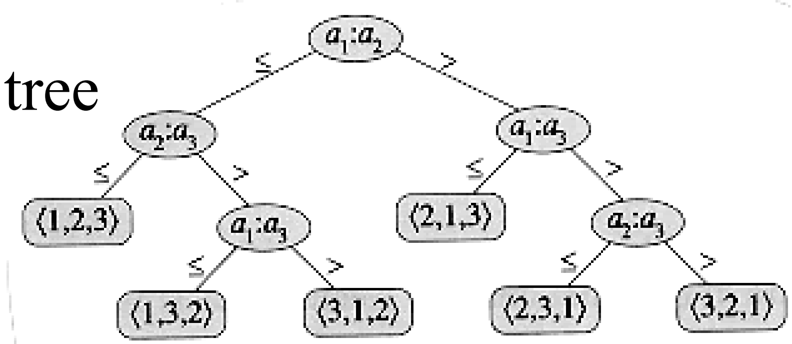
  $\leq$ worst-case number of operations of algorithm

Binary Decision Trees

- Assume elements are the (distinct) numbers 1 through n.
- There are $n!$ leaves (one for each of the $n!$ permutations).
- A binary tree of height h has at most 2^h leaves.

Putting everything together, we get

$$n! \leq \text{number of leaves of decision tree}$$
$$\leq 2^h$$



Thus $h \geq \log_2(n!)$. We have,

$$\log_2(n!) = \log_2(1.\ 2.\ 3.\ \ldots\ .\ n)$$
$$= \log_2 1 + \log_2 2 + \log_2 3 + \ldots \log_2(n/2) + \ldots + \log_2 n$$
$$\geq \log_2(n/2) + \log_2(n/2+1) + \ldots + \log_2 n$$
$$\geq \underbrace{\log_2(n/2) + \log_2(n/2) + \ldots + \log_2(n/2)}_{n/2 \text{ times}}$$
$$= (n/2)\log_2(n/2) = \Omega(n \log n)$$

Then, $h = \Omega(n \log n)$