1 Can we sort by comparisons faster than $\Theta(n \log n)$?  

We have seen two sorting algorithms that run in $\Theta(n \log n)$ time in the worst case. Can we do any better?  

If the algorithm is based on comparisons between array elements, then the answer is: No.  

(Chapter 8 describes a few sorting algorithms not based on comparisons that can be faster.)

2 Decision trees  

Given a comparison-based algorithm and an input size $n$, we can build a decision tree.  

- **Internal nodes** are labeled with comparisons.  
- **Edges** show the algorithm’s progress based on the results of each comparison.  
- **Leaves** represent final permutations of the elements.  
- Other algorithm details are not shown in the decision tree.  

Key idea: The height of the tree gives a lower bound on the run time of the algorithm.  

3 Example: Sorting for $n = 3$  

4 How short can a sorting decision tree be?  

In any decision tree that correctly sorts $n$ distinct elements:
There must be at least $n!$ leaves, one for each permutation of the $n$ elements.

A binary tree with height $h$ has at most $2^h$ leaves.

Therefore $n! \leq 2^h$, and we have

$$T(n) \geq h \geq \log n! = \log \left( \prod_{i=1}^{n} i \right)$$

$$= \sum_{i=1}^{n} \log i \geq \sum_{i=[n/2]}^{n} \log i$$

$$\geq \sum_{i=[n/2]}^{n} \log \left[ \frac{n}{2} \right] \geq \left( \frac{n}{2} \right) \log \left( \frac{n}{2} \right)$$

$$= \Omega(n \log n)$$

Therefore: Any correct comparison-based sorting algorithm takes $\Omega(n \log n)$ time.