Sorting Lower Bound

Comparison Based Sorting

Recall - Sorting

- input: A sequence of n values $x_1, x_2, ..., x_n$
- output: A permutation $y_1, y_2, ..., y_n$ such that $y_1 \leq y_2 \leq ... \leq y_n$

Many algorithms are comparison based

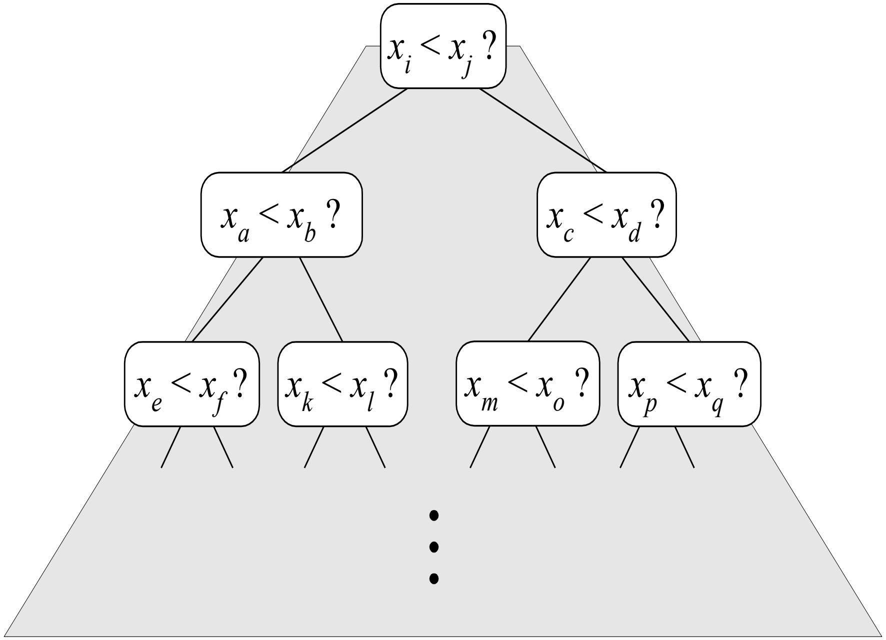
- they sort by making comparisons between pairs of objects
- ex: selection-sort, insertion-sort, heap-sort, merge-sort, quick-sort, …
- best so far runs in $O(n\log n)$ time… can we do better?

Let's derive a lower bound on the running time of any algorithm that uses comparisons to sort n elements $x_1, x_2, ..., x_n$

Counting Comparisons

A decision tree represents every sequence of comparisons that an algorithm might make on an input of size n

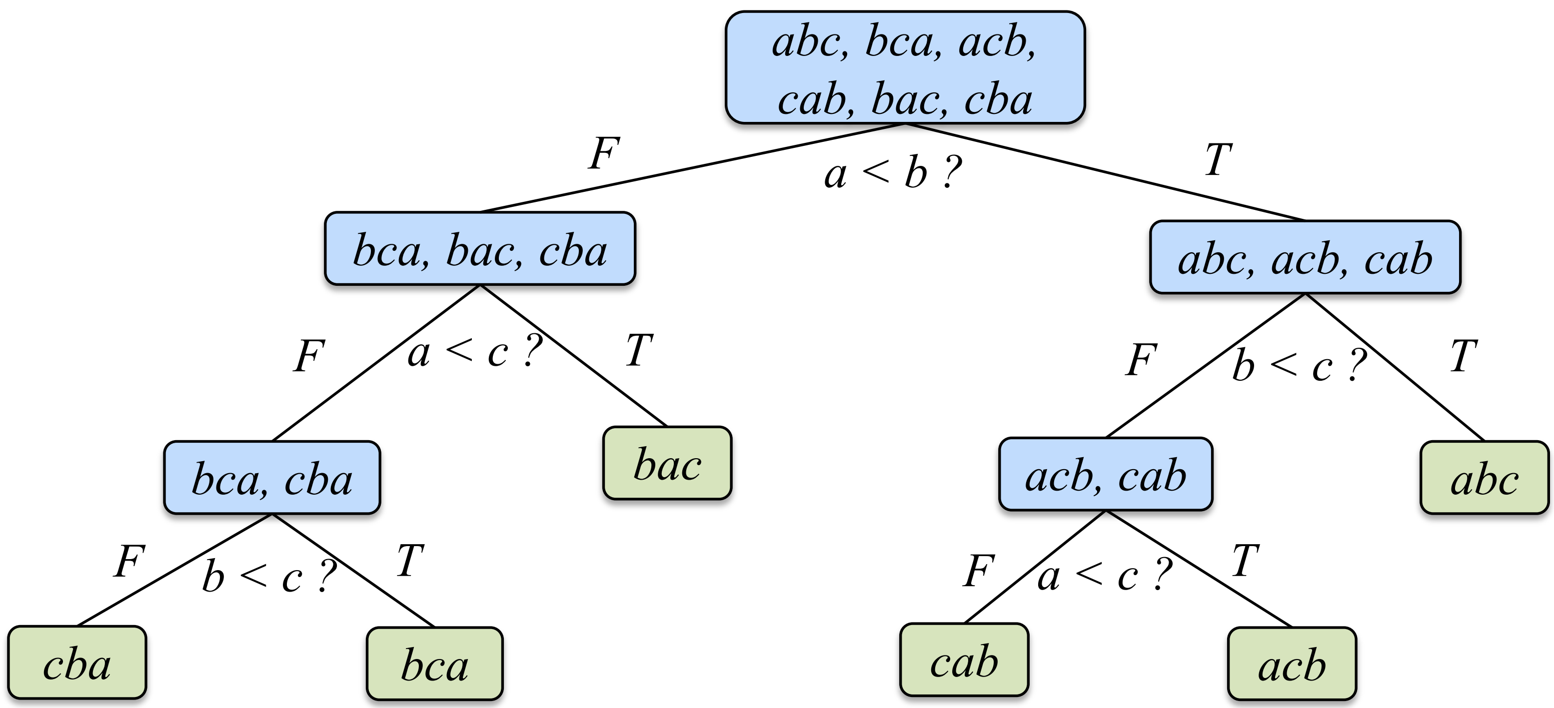
- each possible run of the algorithm corresponds to a root-to-leaf path
- at each internal node a comparison $x_i < x_j$ is performed and branching made
- nodes annotated with the orderings consistent with the comparisons made so far
- leaf contains result of computation (a total order of elements)



Decision Tree Example

Algorithm: insertion sort

Instance ($n = 3$): the numbers a, b, c



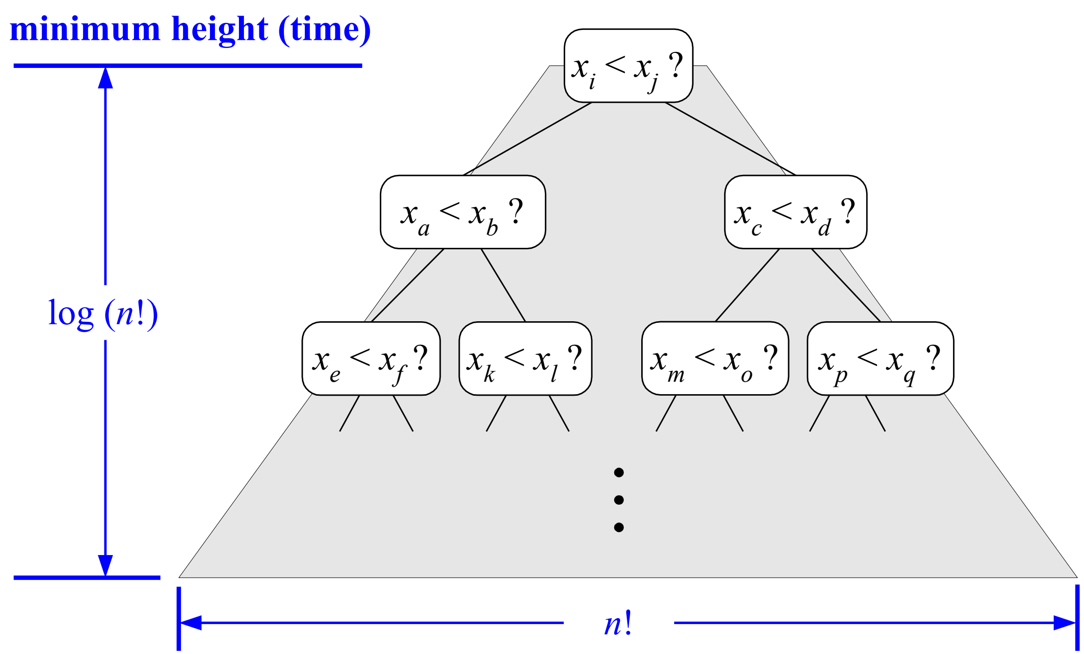
Height of a Decision Tree

Claim: The height of a decision tree is $\Omega(n\log n)$.

Proof: There are $n!$ leaves. A tree of height h has at most 2^h leaves. So

$$\begin{aligned} 2^h &\geq n! \\ h &\geq \log_2(n!) \\ &\geq c\cdot\log_2(n^n) \\ &= c\cdot n\log_2 n. \end{aligned}$$

Thus, $h \in \Omega(n\log n)$.



Lower Bound

Theorem: Every comparison sort requires $\Omega(n\log n)$ in the worst-case.

Proof: Given a comparison sort, we look at the decision tree it generates on an input of size n.

- Each path from root to leaf is one possible sequence of comparisons
- Length of the path is the number of comparisons for that instance
- Height of the tree is the worst-case path length (number of comparisons)

Height of the tree is $\Omega(n\log n)$ by the previous claim. Hence, every comparison sort requires $\Omega(n\log n)$ comparisons.