CSC 226

Algorithms and Data Structures: II

Rich Little
rlittle@uvic.ca
ECS 516

How fast can we sort? A lower bound for comparison based sorting

- We prove: using a comparison based sorting algorithm, we cannot do better than O(n log(n)) worst case time complexity

- Therefore, **Ω(*n* log(*n*))** denotes the **lower bound** for comparison based sorting

Theorem: No comparison based sorting algorithm for n distinct elements has a worst case running time that is better than $O(n \log n)$

Proof:

- Consider a sequence S containing n distinct elements, say $x_0, x_1, x_2, \ldots, x_{n-1}$
- To decide the order of elements, a comparison-based algorithm compares elements pairwise—a sufficient number of times
- In particular, to decide which element of x_i and x_j is smaller, it answers "is $x_i < x_j$?"
- Depending on the outcome—i.e., yes or no—the algorithm performs either no further comparisons or it continues with more comparisons

Proof (continued)

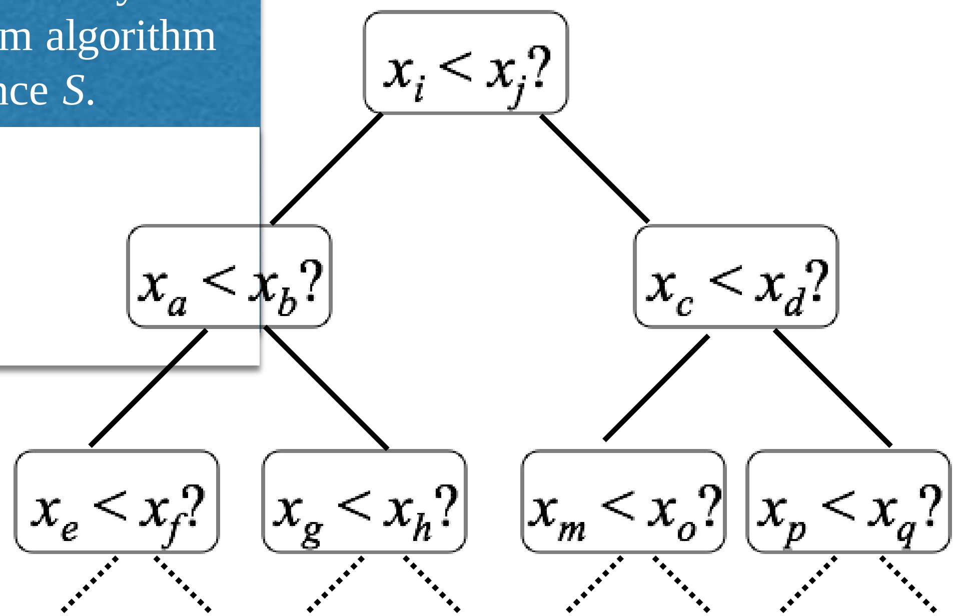
- We want to know: how good is the best of all comparison-based sorting algorithms? (Let’s call it the *optimal* algorithm)
- This optimal sorting algorithm requires a certain number of comparisons (at least) to sort *any* sequence (not just the easiest input)
- We ask: How many comparisons are required for an optimal sorting algorithm to sort *n* elements?

Proof (continued)

- How many comparisons are required for an optimal sorting algorithm to sort n elements?
- We consider our sequence S: x_0, x_1, x_2,…, $x_{n\text{-}1}$
- The optimal algorithm will pick two elements for a first comparison, and, based on the outcome choose a second, etc.
- This can be depicted in a decision tree

Decision tree of an optimal sorting algorithm that is sorting a general sequence of elements

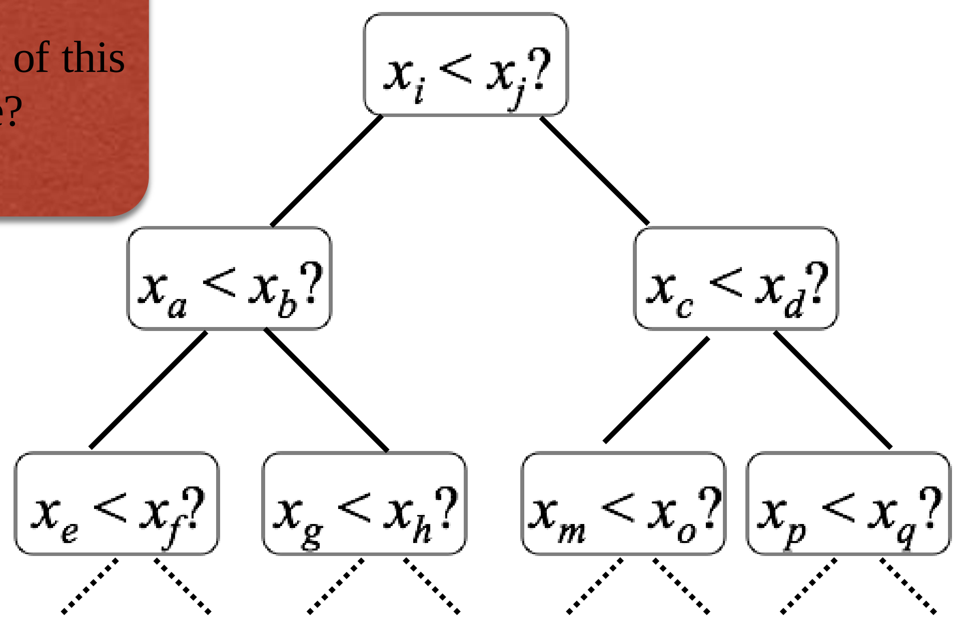
- The decision tree contains every possible path the optimum algorithm might take to sort sequence *S*.
- How many leaves does the tree have at least?



Since we don’t know what *S* looks like, any permutation of *S* could be the sorted one. Thus, every permutation of *S* has to be represented by a path from the root to a leaf in the decision tree. Therefore:

Decision tree of an optimal sorting algorithm sorting a general sequence of elements

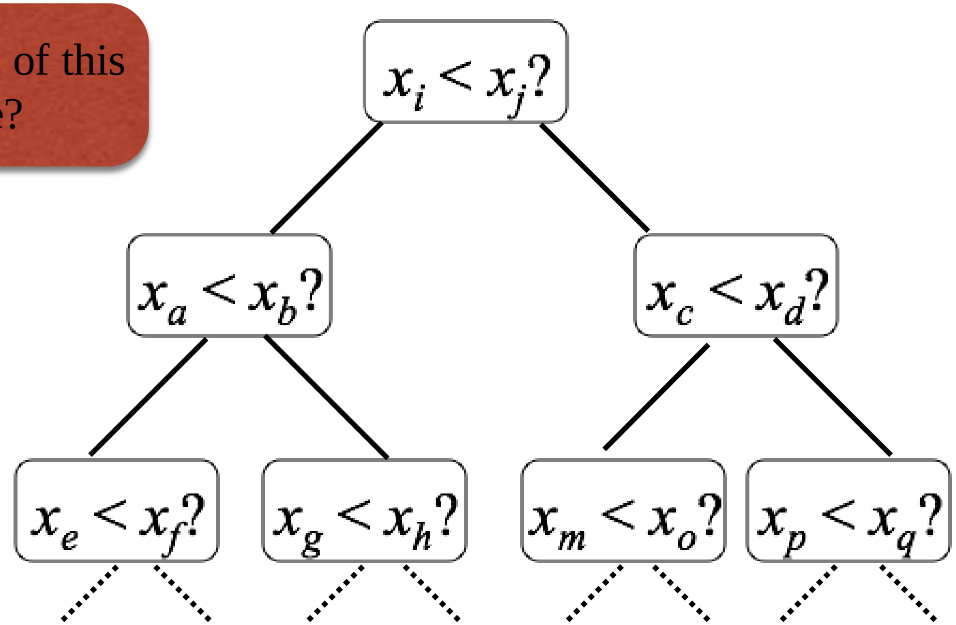
What is the height of this decision tree?



leaves $\geqq n!$

Decision tree of an optimal sorting algorithm sorting a general sequence of elements

What is the height of this decision tree?



- # leaves ≧ n!
- tree is binary
- height is ≧ log(n!); can't be less, since the shortest binary tree that has n! leaves has a height of log(n!) [definition of log]

Proof (continued)

- Since the height of the tree is at least log(n!), we know that
  - at least log(n!) worst case comparisons are required by an optimal comparison based sorting algorithm
  - at least log(n!) worst case comparisons are required by any comparison based algorithm

Proof (continued)

- What is $\Omega(\log(n!))$?

- $\log(n!) \geqq \log(n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1)$
  $\geqq \log(n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n/2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1)$
  $\geqq \log((n/2) \cdot \ldots \cdot (n/2))$
  $\geqq \log((n/2)^{(n/2)}) = (n/2) \log (n/2)$

- and therefore $\log(n!) \in \Omega(n \log(n))$

- We conclude: there is no comparison-based sorting algorithm that has a worst-case time complexity that is better than $O(n \log(n))$