Lower Bounds for Sorting

Adapted from the CLRS book slides

OVERVIEW

How fast can we sort?

- We will prove a lower bound, then beat it by playing a different game.

- **Comparison sorting**
  - All sorts seen so far are comparison sorts: insertion sort, merge sort, quicksort, heapsort, etc.
  - The only operation that may be used to gain order information about a sequence is comparison of pairs of elements.

LOWER BOUNDS FOR SORTING

Lower bounds

- $\Omega(n)$ to examine all the input.
- All sorts seen so far are $\Omega(n \lg n)$.
- We'll show that $\Omega(n \lg n)$ is a lower bound for comparison sorts.

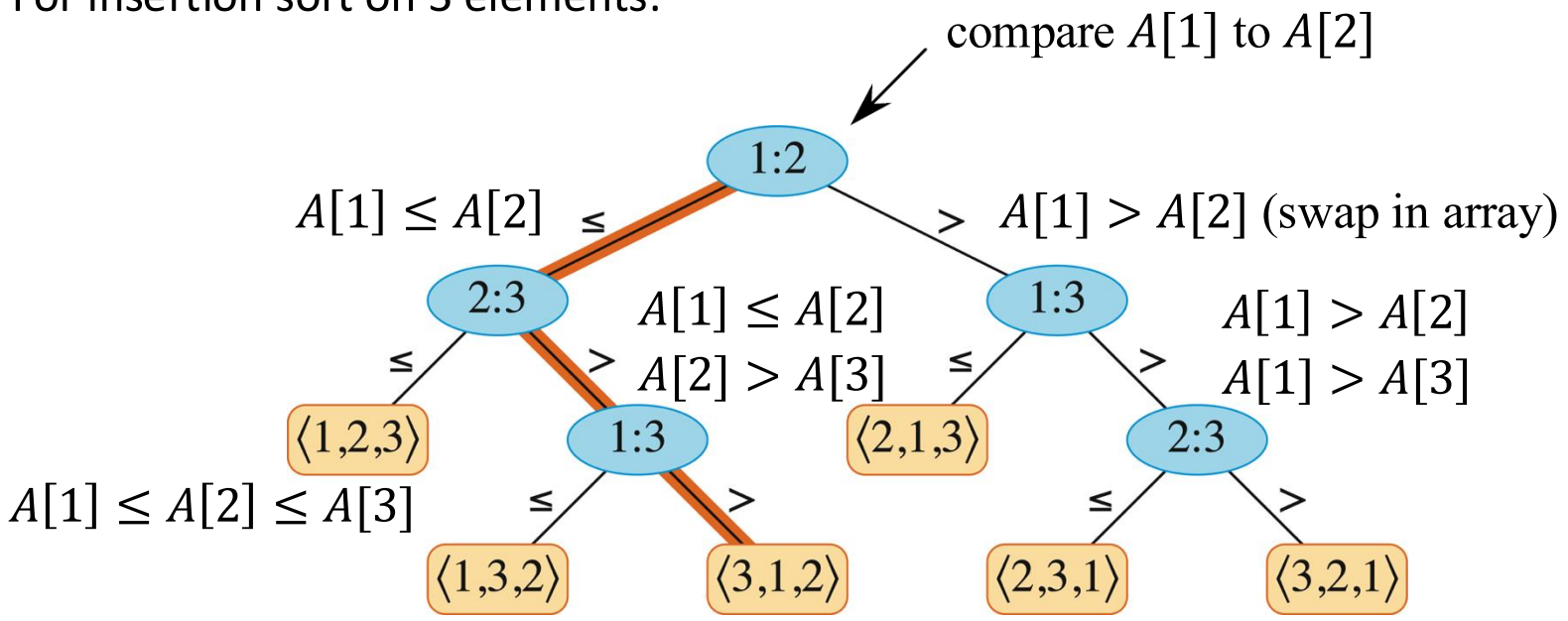
LOWER BOUNDS FOR SORTING (continued)

Decision tree

- Abstraction of any comparison sort.
- Represents comparisons made by a specific sorting algorithm on inputs of a given size.
- Abstracts away everything else: control and data movement.
- We're counting ***only*** comparisons.

LOWER BOUNDS FOR SORTING (continued)

For insertion sort on 3 elements:



How many leaves on the decision tree? There are $\geq n!$ leaves, because every permutation appears at least once.

LOWER BOUNDS FOR SORTING (continued)

For any comparison sort,

1 tree for each n.

Each root-leaf path models a possible execution trace; each node encodes the information determined up to that point.

The tree models all possible execution traces.

What is the length of the longest path from root to leaf?

Depends on the algorithm

Insertion sort: $\Theta(n^2)$

Merge sort: $\Theta(n \lg n)$

THEOREM

Any decision tree that sorts n elements has height $\Omega(n \lg n)$.

Proof Let the decision tree have height h and l reachable leaves.

- $l \geq n!$
- $l \leq 2^h$ (see Section B.5.3: in a complete k-ary tree, there are k^h leaves)
- $n! \leq l \leq 2^h$ or $2^h \geq n!$
- Take logarithms: $h \geq \lg(n!)$
- Use Stirling's approximation: $n! > (n/e)^n$

$$\begin{aligned} h &\geq \lg(n/e)^n \\ &= n \lg(n/e) \\ &= n \lg n - n \lg e \\ &= \Omega(n \lg n) \ . \end{aligned}$$