1. Outline of Today’s Class

2. Lower Bound on Comparison-based Sorting
   - Decision Trees
How Fast Can We Sort?

- The sorting algorithms we have seen so far are insertion sort, mergesort, heapsort, and quicksort.
- All these sorting algorithms are comparison sorts.
- They rely on comparisons to determine the relative order of elements.
- The best worst-case running time that we have seen for comparison sorting is $O(n \cdot \log n)$.
- Is $O(n \cdot \log n)$ the best we can do?
- We need to employ decision trees to answer this question.
Reason for Employing a Decision Tree

Sort \(\langle a_1, a_2, \ldots, a_n\rangle\):

Each internal node is labeled \(i:j\) for \(i,j \in \{1, 2, \ldots, n\}\)

- The left subtree shows subsequent comparisons if \(a_i \leq a_j\)
- The right subtree shows subsequent comparisons if \(a_i > a_j\)
Sort $\langle a_1, a_2, \ldots, a_n \rangle = < 9, 4, 6 >$:

Each internal node is labeled $i : j$ for $i, j \in \{1, 2, \ldots, n\}$

- The left subtree shows subsequent comparisons if $a_i \leq a_j$
- The right subtree shows subsequent comparisons if $a_i > a_j$
Example of a Decision Tree

Sort $\langle a_1, a_2, \ldots, a_n \rangle = \langle 9, 4, 6 \rangle$:

Each internal node is labeled $i : j$ for $i, j \in \{1, 2, \ldots, n\}$

- The left subtree shows subsequent comparisons if $a_i \leq a_j$
- The right subtree shows subsequent comparisons if $a_i > a_j$
Sort $\langle a_1, a_2, \ldots, a_n \rangle = \langle 9, 4, 6 \rangle$:

Each internal node is labeled $i : j$ for $i, j \in \{1, 2, \ldots, n\}$

- The left subtree shows subsequent comparisons if $a_i \leq a_j$
- The right subtree shows subsequent comparisons if $a_i > a_j$
Sort \( \langle a_1, a_2, \ldots, a_n \rangle = \langle 9, 4, 6 \rangle \):

Each leaf contains a permutation \( \langle \pi(1), \pi(2), \ldots, \pi(n) \rangle \) which establishes the ordering \( a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)} \)
A decision tree can model the execution of any comparison sort:

- One tree for each input size $n$
- View the algorithm as splitting the tree whenever it compares two elements
- The tree contains the comparisons along all possible instruction traces
- The running time of the algorithm is then the length of the actual path taken
- Worst-case running time is the height of tree
**Theorem:** Any decision tree that can sort $n$ elements must have height $\Omega(n \cdot \lg n)$

**Proof:**

The tree must contain $\geq n!$ leaves, since there are $n!$ possible permutations.

A height $h$ binary tree has $\leq 2^h$ leaves

Hence, $n! \leq 2^h$

$$h \geq \lg(n!)
\geq \lg((n/e)^n) - \text{Stirling's approximation}
= n \cdot \lg n - n \cdot \lg e
\in \Omega(n \cdot \lg n)$$

**Corollary:** Heapsort and mergesort are asymptotically optimal comparison-based sorting algorithms