Comparison-based lower bound for sorting
(or How fast can we sort?)

This lecture is a little bit abstract; we try to prove the following theorem.

“Any comparison based sorting algorithm takes $\Omega(n \log n)$ to sort a list of $n$ distinct elements in the worst-case.”
“Any comparison based sorting algorithm takes $\Omega(n \log n)$ to sort a list of $n$ distinct elements in the worst-case.”

A comparison based sorting algorithm (comparison sort for short) produces a sorted order which is determined **ONLY** on comparisons (‘>’, ‘<’, ‘\geq’, ‘\leq’, etc.) between the input keys.

E.g., insertion sort, merge sort, quick sort, and heap sort are all comparison based sorting algorithms.
“Any comparison based sorting algorithm takes $\Omega(n \log n)$ to sort a list of $n$ distinct elements in the worst-case.”

- All the comparison-based sorting algorithms that we know of follow the theorem.
- But is that a formal proof? Definitely not, as the theorem states that ANY (EVERY) (not just a particular) comparison based sorting algorithm takes $\Omega(n \log n)$ in the worst case.
New idea is needed to prove the theorem

“Any comparison based sorting algorithm takes $\Omega(n \log n)$ to sort a list of $n$ distinct elements in the worst-case.”

- Our old strategy to analyze the running time of a particular algorithm cannot be used to prove the theorem.

- We need to have a new model to argue the worst-case running time for any comparison based sorting algorithm.

- We use a decision tree model to do the analysis.
Decision Tree Model

- Suppose the input sequence is a *permutation* taken from a set of n distinct numbers:

  \{a_1, a_2, a_3, a_4, \ldots, a_n\}

So there are \(n!\) possible input sequences. The comparison sort should be able to sort all \(n!\) sequences.

- A comparison sort uses comparison between the elements to gain order information about the input sequence, i.e., \(a_i < a_j, a_i > a_j\), etc. Denote this kind of key comparison performed by comparison sort to be \(a_i:a_j\).
Decision Tree Model

- $a_i:a_j$
  A comparison made between element $a_i$ and $a_j$
- Two possible outcomes of a comparison
  $(a_i < a_j)$ or $(a_i > a_j)$
- For any comparison sort, it will perform a sequence of key comparisons to sort the keys. The exact sequence of key comparisons depends on the input keys, and the comparison sort.
- We abstract the sequence of key comparisons performed by the comparison sort in a decision tree.
For example, we trace the sequence of key comparisons of insertion sort on \( <a_1, a_2, a_3> \) by a decision tree.

- **Input**: \( <a_1,a_2,a_3> \)
  - **a1 position in the results will always be before a2**
  - **a1 position in the results will always be after a2**

**Decision Tree Model: insertion sort**

- \( a_1:a_2 \)
  - According to the operation of insertion sort, \( a_1 \) will NOT swap with \( a_2 \). So in the next iteration, \( a_2 \) compares with \( a_3 \).

- \( a_2:a_3 \)
  - According to the operation of insertion sort, \( a_1 \) will swap with \( a_2 \). So in the next iteration, \( a_1 \) compares with \( a_3 \).

- \( a_1:a_3 \)
  - \( <a_2,a_1,a_3> \)
Decision Tree Model: insertion sort

Input: <a₁,a₂,a₃>

The output sorted order is <a₁, a₂, a₃>

The output sorted order is <a₂, a₁, a₃>

The output sorted order is <a₂, a₃, a₁>

The output sorted order is <a₃, a₂, a₁>
The path from the root to a leaf node shows the sequence of key comparisons made to each one of the $n!$ outcomes.

Depending on the values of the input keys, the insertion sort performs different sequence of key comparisons.
Decision Tree Model

- In the insertion sort example, the decision tree reveals all possible key comparison sequences for 3 distinct numbers.
- There are exactly $3! = 6$ possible output sequences.
- Different comparison sorts should generate different decision trees.
- It should be clear that, in theory, we should be able to draw a decision tree for ANY comparison sort algorithm.
Decision Tree Model

- Given a particular input sequence, the path from root to the leaf path traces a particular key comparison sequence performed by that comparison sort.
  - The length of that path represented the number of key comparisons performed by the sorting algorithm.
- When we come to a leaf, the sorting algorithm has determined the sorted order.
- Notice that a correct sorting algorithm should be able to sort EVERY possible output sorted order.
- Since, there are $n!$ possible sorted order, there are $n!$ leaves in the decision tree.
Decision Tree Model

- Given a decision tree, the height of the tree represent the longest length of a root to leaf path.
- It follows the height of the decision tree represents the largest number of key comparisons, which is the worst-case running time of the sorting algorithm.

“Any comparison based sorting algorithm takes $\Omega(n \log n)$ to sort a list of $n$ distinct elements in the worst-case.”

- any comparison sort $\xleftarrow{}$ model by a decision tree
- worst-case running time $\xleftarrow{}$ the height of decision tree

- We are very close to what we want, but how to show the lower bound $\Omega(n \log n)$?
“Any comparison based sorting algorithm takes $\Omega(n \log n)$ to sort a list of $n$ distinct elements in the worst-case.”

- We want to find a lower bound ($\Omega$) on the height of a binary tree that has $n!$ leaves
  → What is the minimum height of a binary tree that has $n!$ leaves?
- The binary tree must be a complete tree (recall the definition of complete tree)
- Hence the minimum (lower bound) height is $\log_2(n!)$
Decision Tree Model

- \( \log_2(n!) \)
  
  \[
  \log_2(n!) = \log_2(n) + \log_2(n-1) + \ldots + \log_2(n/2) + \ldots.
  \]

  \[ \geq \frac{n}{2} \log_2(n/2) = \frac{n}{2} \log_2(n) - \frac{n}{2} \]

  So, \( \log_2(n!) = \Omega(n \log n) \).

- It follows the height of a binary tree which has \( n! \) leaves is \( \Omega(n \log n) \) \( \rightarrow \) worst-case running time is \( \Omega(n \log n) \)

- Putting everything together, we have

  “Any comparison based sorting algorithm takes \( \Omega(n \log n) \) to sort a list of \( n \) distinct elements in the worst-case.”