Comparison Lower Bound of Sorting
(Slides for ESTR2102)

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
We already know that $n$ elements can be sorted in $O(n \log n)$ time. This lecture will prove that the time complexity is optimal for comparison-based algorithms. In other words, every such algorithm must incur $\Omega(n \log n)$ time on at least one input.
There are $n!$ different ways to permute the $n$ elements in the input array $A$.

**Example**

For $n = 3$, 6 permutations:


The goal of sorting is essentially to decide which of the $n!$ permutations is the final sorted order.
Comparison-Based Algorithm

Formally, such an algorithm works by continuously shrinking a pool $P$ of possible permutations.

- At the beginning, $P$ contains all the $n!$ permutations.
- Every comparison allows the algorithm to discard all those permutations in $P$ that are inconsistent with the comparison’s result.
- Eventually, $P$ has only 1 permutation left, which is thus the final sorted order.

In other words, at any moment, all the permutations that remain in $P$ are possible results. The algorithm cannot terminate as long as $|P| \geq 2$. 
In general, each comparison allows us to shrink $P$ to either $P_1$ or $P_2$. 

Yufei Tao

Comparison Lower Bound of Sorting (Slides for ESTR2102)
Comparison-Based Algorithm: The Framework

**Framework**

1. $P \leftarrow $ all the $n!$ permutations of $A$
2. while $|P| > 1$
3. make a comparison between elements $e_1$ and $e_2$
4. if $e_1 < e_2$ then
5. $P \leftarrow P_1$, where $P_1$ is the set of permutations in $P$
   consistent with $e_1 < e_2$
6. else
7. $P \leftarrow P_2$, where $P_2$ is the set of permutations in $P$
   consistent with $e_1 > e_2$
8. return the permutation in $P$

Various algorithms differ in how they implement Step 3.
A Worst-Case Lower Bound

- Note that one of $P_1$ and $P_2$ contains at least half of the permutations in $P$ (i.e., either $|P_1| \geq |P|/2$ or $|P_2| \geq |P|/2$).
- The worst case happens when $P$ always shrinks to the larger set between $P_1$ and $P_2$.
- In this case, the size of $P$ shrinks by at most half after each comparison.
- Hence, the number of comparisons required before $|P|$ decreases to 1 is $\log_2(n!)$.

The next slide shows $\log_2(n!) = \Omega(n \log n)$. 
A Worst-Case Lower Bound

\[
\log_2(n!) = \sum_{i=1}^{n} \log_2 i
\]

\[
\geq \sum_{i=n/2}^{n} \log_2 i
\]

\[
\geq \left(\frac{n}{2}\right) \log_2 \left(\frac{n}{2}\right)
\]

\[
= \Omega(n \log n).
\]

We now conclude that any comparison-based algorithm must incur \(\Omega(n \log n)\) time sorting \(n\) elements in the worst case.