Lower Bound on Comparison-based Sorting

Different sorting algorithms may have different time complexity, how to know whether the running time of an algorithm is **best possible**?

We know of several sorting methods (merge-sort, heapsort) that take time $O(n \log n)$ in the worst case to sort a set of n elements. Can this be improved?

For definiteness, we consider only comparison sorts, which work by comparing pairs of elements. All other logic and moving of elements are ok, but no other use of the elements' values is allowed. The pairs to be compared can be chosen in any way at all. For comparison sorts, it makes sense to count the number of comparisons.

How long does it take to sort? (continued)

It is easier to analyze a slightly different problem.
Consider a list $A = (a_1, a_2, \ldots, a_n)$ of distinct values.

- **Problem 1.** Sort A in increasing order.
- **Problem 2.** Determine which order the elements of A are in.
  For example, the order of $A = (20, 40, 10, 30, 50)$ is $(2, 4, 1, 3, 5)$.

There are $n!$ possible answers given a n-element list.

It is easy to see that these two problems require the same number of comparisons:
Given a solution to one of the two problems, it can be modified to solve the other problem without using more comparisons.

Lower Bound on Sorting

An example: Given a 3-element sequence a_1, a_2, and a_3, sort the sequence in increasing order, there are 3!=6 possible sorted sequences are as follows.

1. $a_1 \leq a_2 \leq a_3$, or its corresponding indices $\langle 1, 2, 3 \rangle$
2. $a_1 \leq a_3 \leq a_2$, or its corresponding indices $\langle 1, 3, 2 \rangle$
3. $a_2 \leq a_1 \leq a_3$, or its corresponding indices $\langle 2, 1, 3 \rangle$
4. $a_2 \leq a_3 \leq a_1$, or its corresponding indices $\langle 2, 3, 1 \rangle$
5. $a_3 \leq a_1 \leq a_2$, or its corresponding indices $\langle 3, 1, 2 \rangle$
6. $a_3 \leq a_2 \leq a_1$, or its corresponding indices $\langle 3, 2, 1 \rangle$

Lower Bound on Sorting (count.)

Let $a_1, a_2, \ldots, a_n$ be n elements to be sorted.

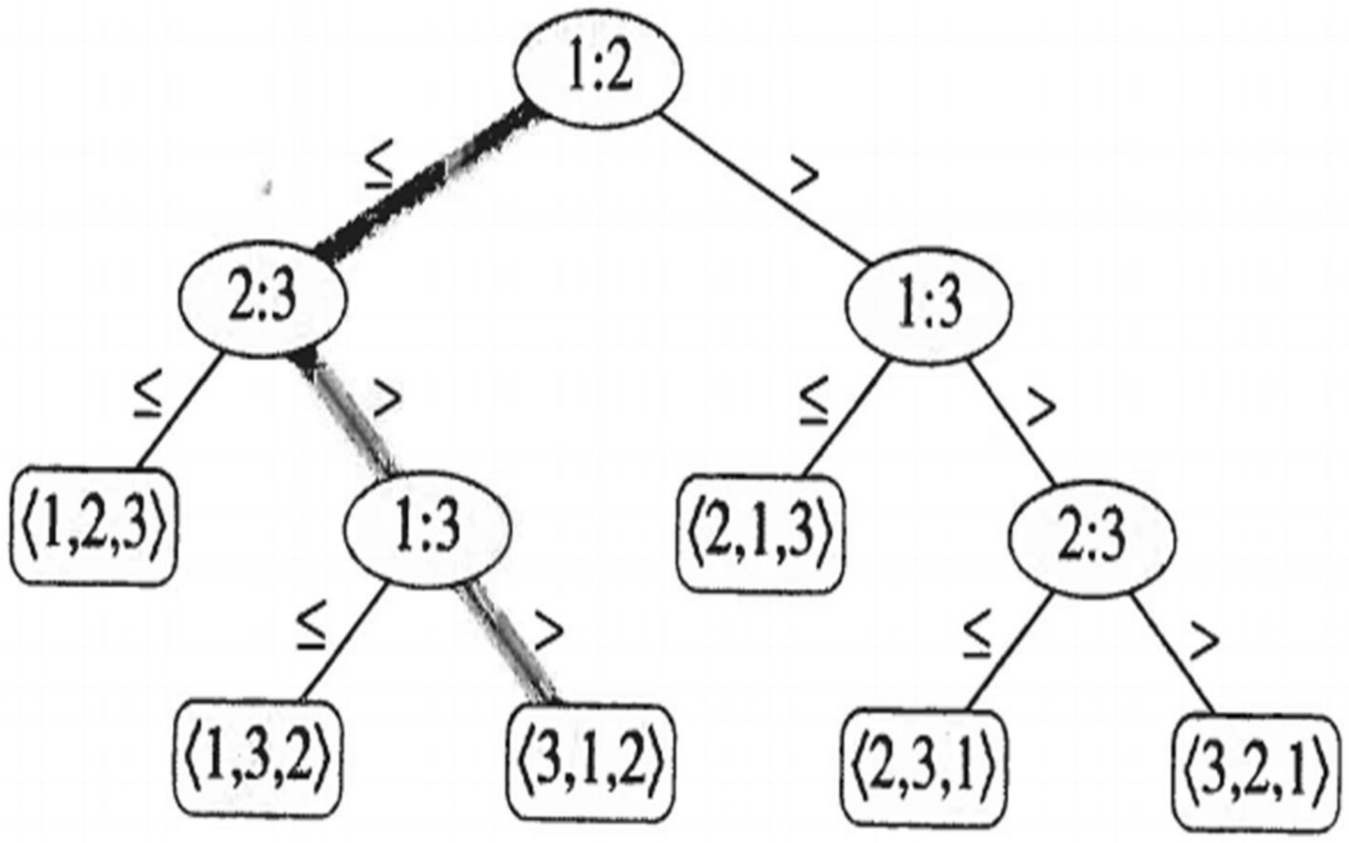
At each time, a pair of elements is compared.

We can use a binary comparison tree to represent the sort procedure, where

- each node in the tree represents one comparison
- each node is labelled by the indices of the pair comparison elements

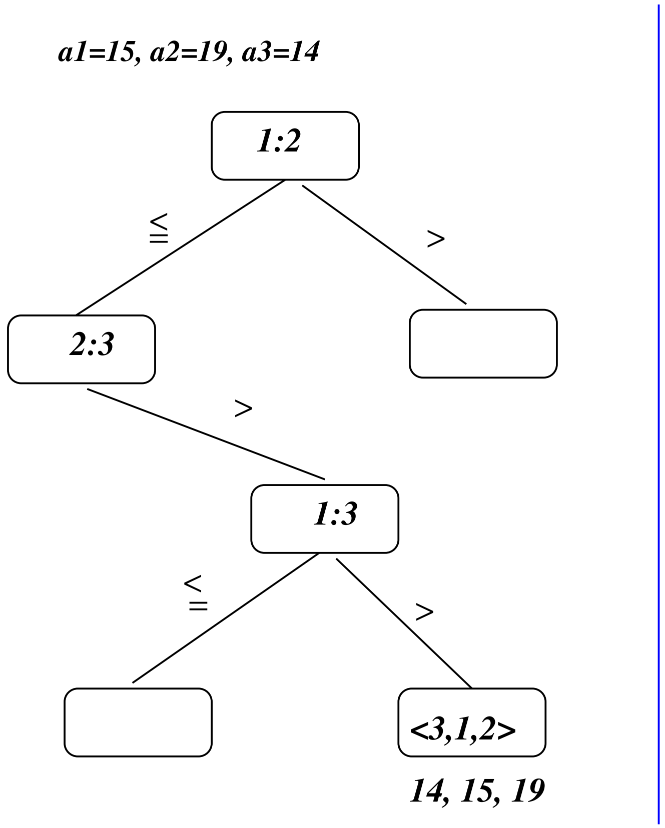
For example, a pair of elements a_1 and a_3 to be compared, we label its corresponding node in the comparison tree as (1 : 3).

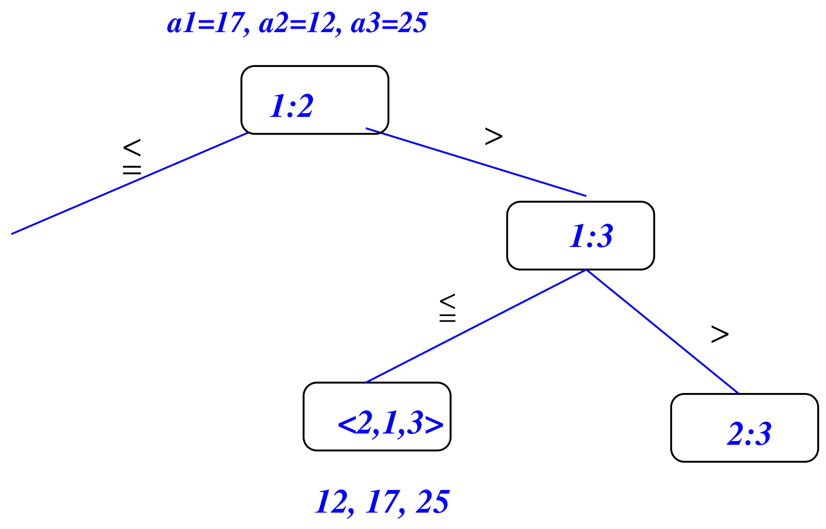
Lower bound on sorting



The input is a sequence of 3 elements: a_1, a_2, a_3.

Lower bound on sorting (examples)





Lower Bound on Sorting

Important observations on the binary comparison tree of sorting:

- The length of a path from the tree root to a tree leaf corresponds to **the number of comparisons to sort a sequence**. The maximum length of any such a path is also referred to as the *the depth* of the comparison tree.
- There are $n!$ tree leaves in the binary comparison tree, as there are $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$ different sorted sequences for n elements to be sorted, depending on the n input element sequence.
- Minimizing the number of comparisons of sorting n elements is equivalent to minimizing the depth (or the height) of the binary comparison tree.

Lower Bound on Sorting

In other words, the problem becomes to construct a binary comparison tree that has $n!$ leaves such that the longest path from the tree root to a tree leaf is minimized.

We observe that each internal node (except the tree root) in the binary comparison tree has a parent node and two children nodes, then,

the depth of any binary tree containing no less than 2^h leaves is at least h, assuming that the depth of the tree root is 0.

Thus, if a binary comparison tree contains $N = n!$ leaves, then its minimum depth h is $\lceil \log N \rceil$, due to the fact that $2^h \geq N$ and $N = n!$.

How long does it take to sort? (continued)

Alternatively, we show the lower bound on sorting by another method.

Suppose we have an algorithm which can always solve **Problem 2** in K comparisons. Each comparison has two possible values “$<$” and “$>$”, assuming that all elements are distinct. Therefore, the sequence of K comparisons has at most 2^K possible values. Such as “$<, >, <, <, >, >, <, <, >, <$”.

Refer to the previous example: $a_1 = 15, a_2 = 19, a_3 = 14$, its corresponding comparison sequence is $a_1 \colon a_2$, $a_2 \colon a_3$, $a_1 \colon a_3$, i.e., “$<, >, >$”.

Since **Problem 2** has $n!$ possible answers, it must be that $2^K \geq n!$. Otherwise, two different answers would have the same sequence of comparison values.
Thus, $K \geq \log(n!)$, and $\log(n!) = \Theta(n \log n)$ by Stirlings formula.

Stirling’s Approximation $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + \Theta(\frac{1}{n}))$.

The Lower bound of the median of n-element sequence

Exercise: Given n distinct elements, show a non-trival lower bound for finding the median of the n elements under the comparison model.

Or Show the following theorem.
Theorem: Any algorithm to find the median of n-elements (for odd n) by comparison of elements must do at least $3n/2 - 3/2$ comparisons in the worst case.