Sorting (VII): The Tight Time Lower Bound of Comparison Based Sorting

CSCD 300 – Data Structures

Eastern Washington University

Goal

We will learn the tight lower bound on the time complexity of any sorting algorithms that are based on the comparisons among the numbers in the input sequence. The lower bound is derived from using the decision tree.

Outline

1. An optimal time lower-bound for comparison-based sorting

An optimal time lower-bound for comparison-based sorting

What is a lower bound of a problem ?

- The cost that is needed by any algorithm (of a specific type) in the worst case.
- The larger is the lower bound we prove, the better is the result we obtain. (So we always want to obtain a lower bound that is as large as possible.)

Our goal

We want to show that any comparison-based sorting algorithm will need at least $\Theta(n \log n)$ time to sort a sequence of n numbers in the worst case. [a]

Or we can equivalently say: We want to show that the time lower-bound of the comparison-based sorting for a sequence of n numbers is $\Omega(n \log n)$. [b]

[a] $\Theta(n \log n)$ can be intuitively understood as a function that is as large as of the order of $n \log n$. The precise definition of Θ will be given in CSCD320.

[b] The precise definition of Ω will be given in CSCD320.

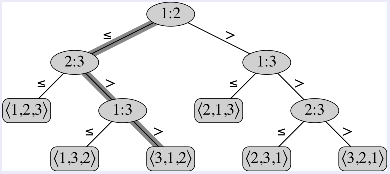
Proof idea: use the decision tree

Decision tree

- At each internal node, we compare two numbers. Based on the result of the comparison, we follow the appropriate child link to go to a low level node, at which we compare another two numbers.
- A leaf node denotes a particular permutation of the input sequence.

An example [a]

[a]Borrowed from Figure 8.1 of *Introduction of Algorithms*, CLRS, 3rd Ed.



- One example decision tree for input sequence $S = \langle 6, 8, 5 \rangle$.
- Node $(i : j)$ means comparing $S[i]$ and $S[j]$.
- A leaf node indicates the sorted version of the input sequence.
- The shaded path are the nodes that we travel for the particular input $S = \langle 6, 8, 5 \rangle$.

Proof idea: use the decision tree (continue)

- Although there can be many different decision trees for an input sequence of size n, every possible decision tree must have exactly $n!$ leaf nodes, representing all possible permutations of n distinct numbers given in the input sequence.
- The number of nodes on any simple path from the root to a leaf is the least number of comparisons we have to make in order to find that sorted sequence represented by that leaf node.
- Because the decision tree is a full binary tree (every internal node has two children) with $n!$ leaves, the height of any decision tree for sequences of size n is at least $\log_2(n!)$.
- That means any comparison-based algorithm has to make at least $\log_2(n!)$ comparisons in the worst case (corresponding to the longest simple path from the root to a leaf).
- $$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n)} \quad \Rightarrow \quad \log_2(n!) = \Theta(n \log n)$$

So, we can conclude:

- Any comparison-based sorting algorithm has to make at least $\Theta(n \log n)$ comparisons in the worst case.

  Or in other words:

- The time lower bound of comparison-based sorting algorithms for sequences of size n is $\Omega(n \log n)$.

Note

- This $\Omega(n \log n)$ lower bound is optimal, meaning we cannot obtain a larger lower bound, because there already exist comparison-based sorting algorithms with $O(n \log n)$ time cost, such as merge sort.
- Merge sort is optimal because its worst-case time cost has matched the lower bound.