lower bound for sorting

Canonical name	LowerBoundForSorting
Date of creation	2013-03-22 12:40:22
Last modified on	2013-03-22 12:40:22
Owner	stevecheng (10074)
Last modified by	stevecheng (10074)
Numerical id	6
Author	stevecheng (10074)
Entry type	Proof
Classification	msc 68P10
Classification	msc 68P30
Related topic	BinaryTree
Related topic	LandauNotation

Several well-known sorting algorithms have average or worst-case running times of $\mathcal{O}(n \log n)$ (heapsort, quicksort). One might ask: is it possible to do better?

The answer to this question is *no*, at least for comparison-based sorting algorithms. To prove this, we must prove that no algorithm can perform better (even algorithms that we do not know!). Often this sort of proof is intractable, but for comparison-based sorting algorithms we can construct a model that corresponds to the entire class of algorithms.

The model that we will use for this proof is a decision tree. The root of the decision tree corresponds to the state of the input of the sorting problem (e.g. an unsorted sequence of values). Each internal node of the decision tree represents such a state, as well as two possible decisions that can be made as a result of examining that state, leading to two new states. The leaf nodes are the final states of the algorithm, which in this case correspond to states where the input list is determined to be sorted. The worst-case running time of an algorithm modelled by a decision tree is the height or depth of that tree.

A sorting algorithm can be thought of as generating some permutation of its input. Since the input can be in any order, every permutation is a possible output. In order for a sorting algorithm to be correct in the general case, it must be possible for that algorithm to generate every possible output. Therefore, in the decision tree representing such an algorithm, there must be one leaf for every one of $n!$ permutations of n input values.

Since each comparison results in one of two responses, the decision tree is a binary tree. A binary tree with $n!$ leaves must have a minimum depth of $\log_2(n!)$. But $\log_2(n!)$ has a lower bound of $\Omega(n \log n)$ (see asymptotic bounds for factorial). Thus any general sorting algorithm has the same lower bound.

This result does not necessarily apply to non-comparison-based sorts, such as the radix sort. Comparison-based sorts – such as heapsort, quicksort, bubble sort, and insertion sort – are more general, in that they depend only upon the assumption that two values can be compared (which is a necessary condition for the sorting problem to be defined for a particular input anyway; see total order). Sorting algorithms such as the radix sort take advantage of special properties that need to hold for the input values in order to reduce the number of comparisons necessary.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithms*, second edition. MIT Press, 2001.