Lower bound on any comparison based sorting algorithm

A comparison based algorithm for sorting, is one which is only allowed to compare pairs of elements of the array for relative value. It is not allowed to inspect the values, the distribution of the input numbers, or how much the numbers differ and so on. We show here that any such algorithm must take $\Omega(n \log n)$ steps to correctly sort an array of $n$ elements, in the worst case. A similar argument, though more involved, can be used to conclude that any such algorithm takes $\Omega(n \log n)$ time on an average. This average case analysis, is based on the assumption that every input permutation is as likely as every other, and holds only in such situations.

We introduce the notion of a computational tree or decision tree to model the progress of an algorithm solving a problem. A single step of computation, in this scenario is a question asked by the algorithm, and the answer being from a fixed predecided set of possibilities. And, the algorithm is a deterministic one, and so runs identically on the same input on multiple runs. Depending on the outcome of the single step of computation, it decides what to do next.

This is modelled as a tree, with the root corresponding to the first step computation performed by the algorithm. The children of the root correspond to the possible second steps of computation. Each corresponds to one of a fixed step of possible outcomes of the first step of the computation. The same procedure continues on levels further down the tree. The leaves correspond to the algorithm having terminated its computation process.

The internal nodes are marked by the computation being performed there, as well as the partial information gathered by the algorithm so far (in its quest to solve the problem). Thus, for the comparison based sorting algorithm, each internal node is marked by comparison between some pair of elements, given by their index within the input array. The internal nodes (and leaves) also contain the partial order of elements whose relative position in the sorted order has been determined. If a pair of elements $a_i, a_j$ are compared, and the indices are such that $i < j$, then the left child corresponds
to the case when \( a_i < a_j \), and the right child corresponds to the case when \( a_i > a_j \). For the sake of simplicity, we assume all the elements in the input array are distinct, but that is justifiable in general by insisting on a stable sorting algorithm. Thus the leaves are marked by the \( n! \) permutations of the input array, as any of these is potentially the correct sorted order.

It follows that this binary computation tree has at least \( n! \) leaves. The computation time on any run is proportional to the length of the path from the root to the corresponding leaf. Thus, the worst case running time corresponds to the depth of the deepest leaf. It is a well known result that any binary tree on \( l \) leaves has height at least \( \log h \). We can conclude, that the computation tree modelling our comparison based sorting algorithm has height at least \( \log(n!) \) which is \( \Theta(n \log n) \). The average case analysis, follows from a proof that the average depth of the leaves in any binary tree on \( l \) leaves is also \( \Omega(\log n) \).

This will be extended, later, to include an other proof of this lower bound based on an adversary argument.