7 Lower bound for sorting

- Five methods of sorting have been mentioned: 'ooo'-sort, quicksort, heapsort, lexical sort, and mergesort. All but one of these methods is based only on comparison of keys (threeway). The exception is lexical sort which is based on analysis of individual keys.
- As a kind of opposite to the $O(g(n))$ notation, there is $\Omega(g(n))$. Write

$$\begin{gathered} f(n) \quad \text{is} \quad \Omega(g(n)) \quad \text{if} \\ (\forall c > 0)(\exists n)(f(n) \geq cg(n)) \end{gathered}$$

- A *comparison-based* sorting method is one whose decisions are based solely on comparing keys.
- We know that mergesort has running time $O(n \log n)$. So has heapsort.
- For any comparison-based sorting method, and any $n > 0$, we can make a binary tree showing how the method works on sets of n keys. This is by 'unrolling' all the for-loops and while-loops so that each such loop is replaced by a series of if-statements. The result is a sorting program for fixed n in the form of a binary tree.
- Given an input in which all keys are distinct (we needn't consider repeated keys), the program traces a path down the tree till it reaches a node where the keys are in sorted order. At that point there is a unique permutation of the input which produces sorted order. Since there are $n!$ possible permutations, the tree contains at least $n!$ nodes. Therefore the depth d of at least one of these nodes satisfies

$$2^{d+1} - 1 \geq n!$$

- There exists at least one input for which the cost of sorting is at least d,

$$d \geq \log_2(1 + n!) - 1 \geq \log_2(n!) - 1$$

- Here is a clever trick to get at a simple version of Stirling's formula.[1] For any k,

$$\begin{gathered} e^n \geq \frac{n^n}{n!} \\ n! \geq \left(\frac{n}{e}\right)^n \\ \ln n! \geq n \ln n - n \ln e \\ \ln n! \geq n(\ln n - 1) \\ d \quad \text{is} \quad \Omega(n \log n) \end{gathered}$$

In other words,

(7.1) Corollary *The worst-case runtime of any* comparison-based *sorting method is* $\Omega(n \log n)$. ∎

So mergesort and heapsort have optimal runtime.

[1] I learnt it a very long time ago from a student, Antony Cutler.