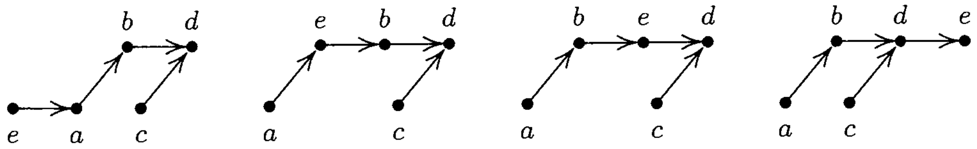
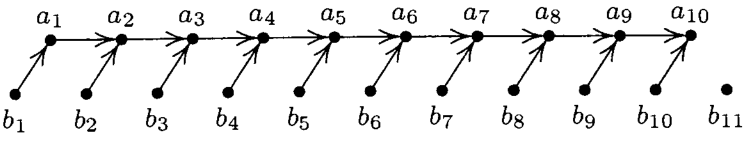
indicating that $a < b < d$ and $c < d$. (It is convenient to represent known ordering relations between elements by drawing directed graphs such as this, where x is known to be less than y if and only if there is a path from x to y in the graph.) At this point we insert the fifth element $K_5 = e$ into its proper place among $\{a, b, d\}$; only two comparisons are needed, since we may compare it first with b and then with a or d. This leaves one of four possibilities,



(6)

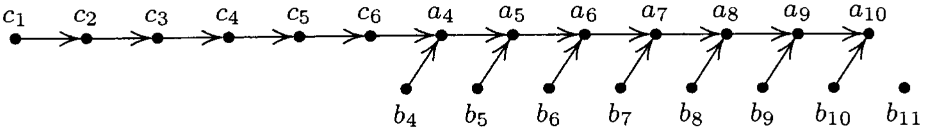
and in each case we can insert c among the remaining elements less than d in two more comparisons. This method for sorting five elements was first found by H. B. Demuth [Ph.D. thesis, Stanford University (1956), 41–43].

Merge insertion. A pleasant generalization of the method above has been discovered by Lester Ford, Jr. and Selmer Johnson. Since it involves some aspects of merging and some aspects of insertion, we shall call it *merge insertion*. For example, consider the problem of sorting 21 elements. We start by comparing the ten pairs $K_1 : K_2, K_3 : K_4, \ldots, K_{19} : K_{20}$; then we sort the ten larger elements of the pairs, using merge insertion. As a result we obtain the configuration


a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11}

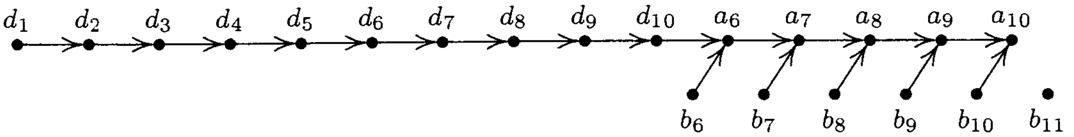
(7)

analogous to (5). The next step is to insert b_3 among $\{b_1, a_1, a_2\}$, then b_2 among the other elements less than a_2; we arrive at the configuration


c_1 c_2 c_3 c_4 c_5 c_6 a_4 a_5 a_6 a_7 a_8 a_9 a_{10}
b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11}

(8)

Let us call the upper-line elements the *main chain*. We can insert b_5 into its proper place in the main chain, using three comparisons (first comparing it to c_4, then c_2 or c_6, etc.); then b_4 can be moved into the main chain in three more steps, leading to


d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} a_6 a_7 a_8 a_9 a_{10}
b_6 b_7 b_8 b_9 b_{10} b_{11}

(9)

The next step is crucial; is it clear what to do? We insert b_{11} (*not* b_7) into the main chain, using only four comparisons. Then b_{10}, b_9, b_8, b_7, b_6 (in this order) can also be inserted into their proper places in the main chain, using at most four comparisons each.

A careful count of the comparisons involved here shows that the 21 elements have been sorted in at most $10 + S(10) + 2 + 2 + 3 + 3 + 4 + 4 + 4 + 4 + 4 + 4 = 66$

steps. Since

$$2^{65} < 21! < 2^{66},$$

we also know that no fewer than 66 would be possible in any event; hence

$$S(21) = 66. \tag{10}$$

(Binary insertion would have required 74 comparisons.)

In general, merge insertion proceeds as follows for n elements:

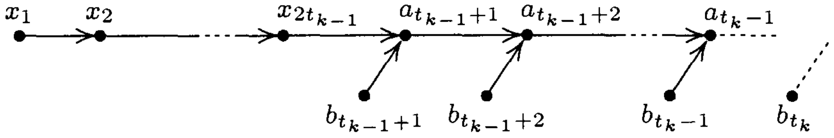
i) Make pairwise comparisons of $\lfloor n/2 \rfloor$ disjoint pairs of elements. (If n is odd, leave one element out.)

ii) Sort the $\lfloor n/2 \rfloor$ larger numbers, found in step (i), by merge insertion.

iii) Name the elements $a_1, a_2, \ldots, a_{\lfloor n/2 \rfloor}, b_1, b_2, \ldots, b_{\lceil n/2 \rceil}$ as in (7), where $a_1 \le a_2 \le \cdots \le a_{\lfloor n/2 \rfloor}$ and $b_i \le a_i$ for $1 \le i \le \lfloor n/2 \rfloor$; call b_1 and the a's the "main chain." Insert the remaining b's into the main chain, using binary insertion, in the following order, leaving out all b_j for $j > \lceil n/2 \rceil$:

$$b_3, b_2;\ b_5, b_4;\ b_{11}, b_{10}, \ldots, b_6;\ \ldots;\ b_{t_k}, b_{t_k - 1}, \ldots, b_{t_{k-1}+1};\ \ldots. \tag{11}$$

We wish to define the sequence $(t_1, t_2, t_3, t_4, \ldots) = (1, 3, 5, 11, \ldots)$, which appears in (11), in such a way that each of $b_{t_k}, b_{t_k - 1}, \ldots, b_{t_{k-1}+1}$ can be inserted into the main chain with at most k comparisons. Generalizing (7), (8), and (9), we obtain the diagram



where the main chain up to and including $a_{t_k - 1}$ contains $2t_{k-1} + (t_k - t_{k-1} - 1)$ elements. This number must be less than 2^k; our best bet is to set it equal to $2^k - 1$, so that

$$t_{k-1} + t_k = 2^k. \tag{12}$$

Since $t_1 = 1$, we may set $t_0 = 1$ for convenience, and we find that

$$\begin{aligned} t_k = 2^k - t_{k-1} = 2^k - 2^{k-1} + t_{k-2} = \cdots &= 2^k - 2^{k-1} + \cdots + (-1)^k 2^0 \\ &= \bigl(2^{k+1} + (-1)^k\bigr)/3 \end{aligned} \tag{13}$$

by summing a geometric series. (Curiously, this same sequence arose in our study of an algorithm for calculating the greatest common divisor of two integers; see exercise 4.5.2–36.)

Let $F(n)$ be the number of comparisons required to sort n elements by merge insertion. Clearly

$$F(n) = \lfloor n/2 \rfloor + F(\lfloor n/2 \rfloor) + G(\lceil n/2 \rceil), \tag{14}$$

where G represents the amount of work involved in step (iii). If $t_{k-1} \le m \le t_k$, we have

$$G(m) = \sum_{j=1}^{k-1} j(t_j - t_{j-1}) + k(m - t_{k-1}) = km - (t_0 + t_1 + \cdots + t_{k-1}), \tag{15}$$

$22!/2^{70} \approx 0.952$ requires extremely high efficiency to sort in 70 steps. (Only 91 of the 1649 graphs found on 12 or fewer vertices had such high efficiency.)

The intermediate results suggest strongly that $S(13) = 33$, so that merge insertion would not be optimum when $n = 13$. It should certainly be possible to prove that $S(16) < F(16)$, since $F(16)$ takes no fewer comparisons than sorting ten elements with $S(10)$ •comparisons and then inserting six others by binary insertion, one at a time. There must be a way to improve upon this! But at present, the smallest case where $F(n)$ is definitely known to be nonoptimum is $n = 47$: After sorting 5 and 42 elements with $F(5) + F(42) = 178$ comparisons, we can merge the results with 22 further comparisons, using a method due to J. Schulte Mönting, *Theoretical Comp. Sci.* **14** (1981), 19–37; this beats $F(47) = 201$. (Glenn K. Manacher [*JACM* **26** (1979), 441–456] had previously proved that infinitely many n exist with $S(n) < F(n)$, starting with $n = 189$.)

The average number of comparisons. So far we have been considering procedures that are best possible in the sense that their worst case isn't bad; in other words, we have looked for "minimax" procedures that minimize the *maximum* number of comparisons. Now let us look for a "minimean" procedure that minimizes the *average* number of comparisons, assuming that the input is random so that each permutation is equally likely.

Consider once again the tree representation of a sorting procedure, as shown in Fig. 34. The average number of comparisons in that tree is

$$\frac{2+3+3+3+3+2}{6} = 2\tfrac{2}{3},$$

averaging over all permutations. In general, the average number of comparisons in a sorting method is the *external path length* of the tree divided by $n!$. (Recall that the external path length is the sum of the distances from the root to each of the external nodes; see Section 2.3.4.5.) It is easy to see from the considerations of Section 2.3.4.5 that the minimum external path length occurs in a binary tree with N external nodes if there are $2^q - N$ external nodes at level $q - 1$ and $2N - 2^q$ at level q, where $q = \lceil \lg N \rceil$. (The root is at level zero.) The minimum external path length is therefore

$$(q-1)(2^q - N) + q(2N - 2^q) = (q+1)N - 2^q. \tag{34}$$

The minimum path length can also be characterized in another interesting way: *An extended binary tree has minimum external path length for a given number of external nodes if and only if there is a number l such that all external nodes appear on levels l and $l+1$.* (See exercise 20.)

If we set $q = \lg N + \theta$, where $0 \le \theta < 1$, the formula for minimum external path length becomes

$$N(\lg N + 1 + \theta - 2^\theta). \tag{35}$$

The function $1 + \theta - 2^\theta$ is shown in Fig. 37; for $0 < \theta < 1$ it is positive but very small, never exceeding

$$1 - (1 + \ln\ln 2)/\ln 2 = 0.08607\ 13320\ 55934+. \tag{36}$$