A minor improvement of Heapsort.

Heapsort is a very efficient algorithm for sorting the elements $M(i)$ for $1 \le i < N$ of a linear array. To sort the elements in ascending order the algorithm maintains $H(p,q)$ defined by

$$H(p,q): \quad (\underline{A}\, i,j:\ p \le i < j < q \wedge 2\cdot i \le j < 2\cdot(i+1):\ M(i) \ge M(j))$$

which enjoys the useful property

$$H(p,q) \wedge p=1 \Rightarrow (\underline{A}\, j: 1 \le j < q:\ M(1) \ge M(j)) \quad . \qquad (0)$$

The algorithm has the following form:

$p, q := (N+1)\ \underline{\text{div}}\ 2, N$; $\{H(p,q)\}$
$\underline{\text{do}}$ $p \ne 1 \rightarrow$ $p := p-1$; $\{H(p+1,q)\}$ sift $\{H(p,q)\}$ $\underline{\text{od}}$;
$\underline{\text{do}}$ $q > 2 \rightarrow$ $q := q-1$; $M{:}swap(1,q)$;
$\{H(p+1,q)\}$ sift $\{H(p,q)\}$
$\underline{\text{od}}$.

Since $p=1$ is a further invariant of the second repetition, property (0) ensures that the sorted list is built up from "right to left."

The routine sift establishes —by $w := p$— and maintains

$$SH: \ (\underline{A}\, i,j:\ p \le i < j < q \wedge 2\cdot i \le j < 2\cdot(i+1):\ M(i) \ge M(j) \vee w=i) \quad ,$$

which enjoys the useful property $SH \wedge 2\cdot w \geq q \Rightarrow H(p,q)$. The routine sift can repeatedly perform under invariance of SH $w := 2\cdot w$ or $w := 2\cdot w+1$; sift compares each time $M(w)$ with the maximum of $M(2\cdot w)$ and $M(2\cdot w+1)$. If $M(w)$ is large enough, $H(p,q)$ holds and sift terminates; otherwise w can be "doubled" at the price of 2 comparisons and 1 swap in array M. For further details we refer the reader to [0].

We can do better by replacing $H(p,q)$ by $H3(p,q)$ — and, similarly, SH by $SH3$ —

$$H3(p,q):\ (\underline{A}\, i,j\colon p \leq i < j < q \wedge 3\cdot i \leq j < 3\cdot(i+1)\colon M(i) \geq M(j)) .$$

Firstly, we can then start with a smaller p, viz. $(N+2)\ \underline{\text{div}}\ 3$; secondly sift can then "triple" w at the cost of 3 comparisons and 1 swap in array M. Now 6 comparisons and 2 swaps multiply w by 9, whereas originally 6 comparisons and 3 swaps were needed for a factor 8. (With the analogous $H4(p,q)$ the gain in comparisons needed is lost again: $2^3 < 3^2$ but $2^4 = 4^2$; the gain at initialization and in number of swaps increases still further. Since $2^5 > 5^2$, $H5(p,q)$ is expected to lead to more comparisons in sift.)

* * *

Some weeks ago I received under the title "Gleanings from Combinatorics" from Ross A. Honsberger of Waterloo University a short series of combinatorial problems with their solutions. One of them was how to partition a positive integer $n \geq 2$ into one or more (positive integer) parts so that the product of the parts is a maximum. The answer is:

for $n = 3k$, use k 3's;
$n = 3k+2$, use k 3's and a 2
$n = 3k+4$, use k 3's and a 4.

(The preponderance of 3's is not so amazing: 3 is the best integer approximation of e, which is the solution of the corresponding continuous problem. The observation $2^3 < 3^2$ was part of the justification of the discrete solution.) That the study of a gem from rather pure mathematics led to the discovery how to improve the efficiency of a standard algorithm, which is justly famous for its efficiency, was a very pleasant surprise. I think it is a lesson.

Plataanstraat 5
5671 AL NUENEN
The Netherlands

8 May 1981

prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow

[0] Niklaus Wirth, Algorithms + Data Structures = Programs, 1976, Prentice-Hall Inc, Englewood Cliffs, USA, pp. 72-76.