An alternative ending for AvG16/EWD809

(Early comments from W.H.J. Feijen and from C.S. Scholten have suggested further disentanglement of smoothsort's presentation.)

In smoothsort the descending tree covering the unsorted prefix is a so-called "leftward tree", i.e. a tree in which each son is situated to the left of its father. As a result, the rightmost element of the unsorted prefix is the root of the tree and hence dominates all other elements of the unsorted prefix. Note that leftward trees are the only ones permissible under the constraint that smoothsort leaves an initially increasing sequence unchanged all through the computation. This constraint is strongly suggested by the aim that smoothsort be best-case of order N ; in a moment we shall encounter a further consequence of that aim.

Its rightmost element dominating all its others, the unsorted prefix can be shortened by one element without violation of P4 . In contrast to heapsort's tree, which is pruned leaf by leaf, smoothsort's tree is pruned in the second phase at its root: it becomes a forest of as many trees as the removed root had sons. Two such trees can be made into a single descending leftward tree by grafting the one with the leftmost

root upon the root of the other (sift being applied in circumstance c); hence the forest can be rebuilt —possibly in many ways— into a single descending leftward tree.

In smoothsort this freedom is exploited to bound the maximum number of sons with the same father, because such an upper bound is a sufficient condition for the number of comparisons to be of order $N \cdot \log N$ in the worst case and of order N in the best case. In smoothsort each father has in fact at most 3 sons. In order to maintain that state of affairs, of the trees of the aforementioned forest only the tree with the leftmost root may have nodes with as many as three sons: by grafting in the order from left to right upon the remaining roots no father with more than 3 sons is introduced. The requirement that no different forest ever emerges implies that all fathers with three sons are on the leftmost path from the root of the covering leftward tree and that the unordered prefix itself is that tree's postorder traversal. (The postorder traversal of a tree is a special permutation of its vertices, viz. the concatenation of the postorder traversals of its first-generation subtrees followed by its root.)

As a result we can —and shall do so in the sequel— view the unsorted prefix as the concatena-

tion of the postorder traversals of one or more binary trees such that, firstly, in each binary tree each father dominates its offspring and, secondly, the roots of the binary trees are ascending in the order in which they occur.

In contrast to heapsort, which permanently uses (part of) the same simple tree, smoothsort has to keep track of the shape of a changing tree, i.e. it has to keep track of the sequence of binary trees whose concatenated postorder traversals form the unordered prefix. The sequence has to be such that the clerical labour involved herein can also be of order N in the best case.

To this purpose, the binary trees admitted are the so-called Leonardo trees LT_i: LT_0 and LT_1 both consist of a single leaf, LT_{i+2} has LT_{i+1} as its left subtree and LT_i as its right subtree. The concatenation is a so-called standard concatenation, i.e. it consists of the postorder traversal of the largest possible Leonardo tree, followed by the standard concatenation for the remainder of the unsorted prefix. Thus we achieve that the unsorted prefix is covered by the minimum number of Leonardo trees. (Leonardo trees have been preferred to balanced binary trees because, on the average, 25% more trees are needed for coverage by the latter.)

As in heapsort, smoothsort's second phase —in which the sorted sequence is built up from the right— is preceded by a first phase in which the unsorted prefix of length N (i.e. covering the whole sequence) is prepared. In contrast to heapsort, this preparation starts from the left; q —in the first phase the length of the prepared prefix— is initialized at 1 and repeatedly increased by 1 until q = N.

The first phase's main task is to see to it that at each increase of q the Leonardo trees covering the prefix are such that each (binary) father dominates its (binary) offspring. There are two cases. If increasing q by 1 boils down to extending the standard concatenation by a one-node Leonardo tree, this obligation is empty; otherwise LT_{i+1}, LT_i, and the new element are combined into LT_{i+2}, to whose root sift is applied (circumstance b).

The first phase's second obligation is to insert a grafting operation (circumstance c) each time a Leonardo tree is formed that will not subsequently be absorbed in a larger Leonardo tree; as a result, at the end of the first phase the roots of the binary trees are in ascending order.

In order to arrive at an algorithm of order N

in the best case, a stack records which Leonardo trees cover the unsorted prefix. (Because each Leonardo tree occurs at most once in a standard concatenation, a bit stack in fact suffices.)

12 February 1982

drs. A.J.M. van Gasteren
BP Venture Research Fellow
Dept. of Mathematics and
Computing Science
University of Technology
5600 MB EINDHOVEN
The Netherlands

prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow
Plataanstraat 5
5671 AL NUENEN
The Netherlands