JOURNAL OF COMPUTER AND SYSTEM SCIENCES 13, 184–199 (1976)

Finding the Median*

A. SCHÖNHAGE

Department of Mathematics, University of Tübingen, Germany

M. PATERSON

Department of Computer Science, University of Warwick, Coventry, United Kingdom

N. PIPPENGER

Mathematical Sciences Department, IBM Research Center, Yorktown Heights, New York

Received November 7, 1975; revised April 9, 1976

An algorithm is described which determines the median of n elements using in the worst case a number of comparisons asymptotic to $3n$.

1. INTRODUCTION

For many problems concerned with ordering elements there is only a narrow gap between lower bounds on the number of pair-wise comparisons required in the worst case and upper bounds provided by simple algorithms. For the problem of sorting n totally ordered elements, for example, upper and lower bounds asymptotic to $n\log_2 n$ are easily obtained (see [3, Sect. 5.3.1]). In contrast with this, the problem of finding the median (or, when n is even, either of the two "medians") has proved much more challenging, and only recently has any upper bound that is $O(n)$ been obtained. At present the best lower bound known is $1.75n + O(n)$ [2, 4], while the best upper bound previously reported is $5.43n + o(n)$ ([1], $5.43\cdots = 391/72$). In this paper we shall improve this upper bound by giving an algorithm using only $3n + o(n)$ comparisons.

We may always assume that n is odd, for if n is even we can adjoin a "$+\infty$" or "$-\infty$" to the elements to compute either of the two "medians."

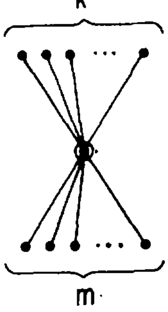
* This research was supported in part by the National Science Foundation under research grant GJ-34671 to MIT Project MAC.

2. Theoretical Framework

We suppose that we are given a totally ordered set T. The order is not known initially and can only be determined by performing a sequence of pair-wise comparisons between elements of T. After any such sequence, the current information about the order of T is expressible as a partial order Q on the elements of T.

Let P be a partially ordered set. A sequence of comparisons between elements of T is said to *produce* P in T when the partial order Q obtained "contains" P, that is, when there is an order-preserving embedding of P into Q. If $|T| = n \geqslant |P|$, we define $g(P, n)$ to be the minimum number of comparisons required in the worst case by an optimal algorithm to produce P in T, and let $g(P)$ denote $g(P, |P|)$.

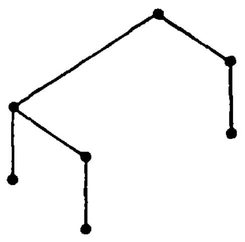
The partial order $S_m{}^k$ on $k + m + 1$ elements has one particular element, the *center*, which is less than each of k other elements and greater than each of the m remaining elements. Partial orders will be depicted by their Hasse diagrams, thus $S_m{}^k$ by



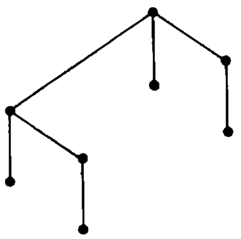
The problem of finding the median of n elements is the problem of producing $S_k{}^k$ in T when $|T| = 2k + 1 = n$. In this paper we shall derive an upper bound for $g(S_k{}^k)$.

3. Yao's Hypothesis

It is clear that $g(P, n)$ is a decreasing function of n, since one can always ignore any extra elements. That $g(P, n)$ can strictly decrease is shown when P is



for then $g(P, 7) = 8$ (as can be shown by an enumeration of cases) but $g(P, 8) = 7$, since P can be embedded in the partial order



which is produced using 7 symmetric comparisons.

Yao conjectured in [5] that $g(S_m{}^k, n) = g(S_m{}^k)$ for all $n \geqslant k + m + 1$ or, in words, that extra elements do not help in producing $S_m{}^k$. We know of no counter example, and this hypothesis has a remarkable consequence that stimulated the development of our algorithm.

THEOREM 3.1. *(Proved in* [5] *for a bound of* $3n$.) *Under Yao's hypothesis, there is a median algorithm for n elements that uses at most* $2.5n + o(n)$ *comparisons.*

Proof. Take $4k + 2$ elements and perform $2k + 1$ comparisons to obtain $2k + 1$ disjoint pairs. Apply to the lower $2k + 1$ elements an optimal algorithm that produces $S_k{}^k$. The result contains the partial order shown in Fig. 1, which contains S_k^{2k+1}.

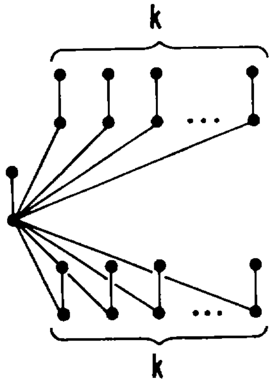


FIGURE 1

Thus by Yao's hypothesis

$$\begin{aligned} g(S_k^{2k+1}) &= g(S_k^{2k+1}, 4k + 2) \\ &\leqslant 2k + 1 + g(S_k{}^k). \end{aligned}$$

Similarly take $6k + 4$ elements and perform $3k + 2$ comparisons to obtain $3k + 2$ disjoint pairs. Apply an optimal algorithm that produces S_k^{2k+1} to the upper $3k + 2$ elements. The result contains S_{2k+1}^{2k+1}. Thus by Yao's hypothesis.

$$\begin{aligned} g(S_{2k+1}^{2k+1}) &= g(S_{2k+1}^{2k+1}, 6k + 4) \\ &\leqslant 3k + 2 + g(S_k^{2k+1}). \end{aligned}$$

Combining these results we obtain

$$g(S_{2k+1}^{2k+1}) \leqslant 5k + 3 + g(S_k{}^k).$$

Since $g(S_0{}^0) = 0$ and $g(S_{2k}^{2k}) \leqslant g(S_{2k+1}^{2k+1})$, an iteration of this inequality yields

$$g(S_k{}^k) \leqslant 5k + O(\log k)$$

Setting $n = 2k + 1$ gives the theorem. ▌

The remainder of this paper will be devoted to proving (without using Yao's hypothesis) that there is a median algorithm for n elements that uses at most $3n + o(n)$ comparisons.

4. Factory Production

In general to produce m disjoint copies of a partial order P, which we denote by $m \times P$, may require fewer than m times the number of comparisons to produce a single P. For example, $g(S_3{}^1, k) = 6$ for $k \geqslant 5$, but $g(2 \times S_3{}^1) \leqslant 11$. A systematic exploitation of this sort of economy is achieved by "factories." We shall sustain the metaphor in which we regard the developping partial order as a graph which is being "assembled" from component parts by making new comparisons.

A *factory* for a partial order P is a comparison algorithm with continual input and output of elements. The input, in the simplest kind of factory, consists of singleton elements. At intervals a new disjoint copy of P is output. The vital characteristics of a factory are the number of comparisons needed to set up the factory, the *initial cost* I, the number of comparisons then needed to produce each copy of P, the *unit cost* U, and thirdly the *production residue* R, which is the maximum number of elements that can remain in the factory when lack of input stops production. For all $m \geqslant 0$, $I + m \cdot U$ gives an upper bound on the number of comparisons required to produce $m \times P$.

An unexpected feature of our median algorithms is that they involve the factory production of many copies of $S_k{}^k$ for some k, where $k = o(n)$. The following theorem is eventually proved in Section 9. It is stated here to provide motivation for the next section.

THEOREM 4.1. *For any k, there is a factory F_k for S_k^k with characteristics I_k, U_k, R_k as defined above such that*

$$U_k \sim 5k, \qquad I_k = O(k^2), \qquad R_k = O(k^2).$$

Several stronger results will be given in later sections. At this stage we shall explain how such factories can be exploited to provide economical median algorithms.

5. BASIC MEDIAN ALGORITHM

THEOREM 5.1. *Given factories F_k satisfying $U_k \sim Ak$ for some $A > 0$, $I_k = O(k^2)$ and $R_k = O(k^2)$, there is a median algorithm which, for n elements (n odd), uses at most $An + o(n)$ comparisons.*

Proof. Given n, let $k = \lfloor n^{1/4} \rfloor$. The algorithm first sets up the factory F_k using I_k comparisons. The subsequent operations involve four interconnected processes.

(i) Whenever sufficiently many input elements are supplied to F_k a new copy of S_k^k is produced at cost U_k.

(ii) The centers of all the S_k^k which are produced are to be ordered. As each new S_k^k is completed, its center is inserted using binary insertion (see [3]) into an ordered chain of previous centers.

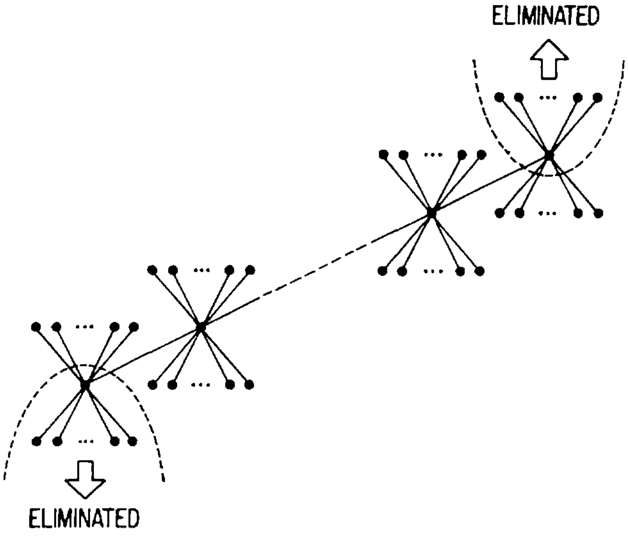


FIGURE 2

(iii) When no further S_k^k can be produced, i.e., (i) has halted, consider the two extreme S_k^k's of the ordered chain (see Fig. 2).

Suppose the chain is of length t and the total number of elements remaining in the factory is r. The center of the upper S_k^k is above each other center and above each element in the bottom half of any S_k^k. *Provided* $t - 1 > r$, this center and all the elements in the upper half of the upper S_k^k are above more than half of the total number of elements. Therefore these $k + 1$ elements are above the median, and similarly, the center and lower half of the lower S_k^k are below the median. All these $2(k + 1)$ elements can be eliminated from the algorithm, leaving the problem of determining the median of the remaining elements. The lower half of the upper S_k^k and the upper half of the lower S_k^k are returned to the factory as $2k$ singleton elements.

(iv) When none of (i), (ii), (iii) can proceed we must have $r \leqslant R_k$ (from (i)) and $t \leqslant r + 1$ from (iii)). Thus the total number m of remaining elements must satisfy

$$m = t(2k + 1) + r \leqslant (R_k + 1)(2k + 1) + R_k = O(k^3)$$

by the conditions of the theorem. The median of these m elements, which is the median of the original set, can be found by any linear median algorithm or, as we prefer here for simplicity, by sorting them using $O(k^3 \cdot \log k)$ comparisons.

The total number of S_k^k's produced in F_k is precisely $(n - m)/(k + 1) + t$, where m, t are the numbers used in (iv). For each, the binary insertion in (ii) requires at most $\lceil \log_2 (n/(2k + 1)) \rceil$ comparisons. The number of comparisons used by the complete algorithm is therefore at most

$$\begin{aligned} I_k + ((n - m)/(k + 1) + t)(U_k + \lceil \log_2 (n/(2k + 1)) \rceil) + O(k^3 \cdot \log k) \\ \leqslant (n/k)U_k + O((n \log n)/k) + O(k^3 \log k) \sim An \end{aligned}$$

since $t \leqslant r + 1 = O(k^2)$, $k \sim n^{1/4}$, and $U_k \sim Ak$ ∎

Using Theorem 4.1 we immediately have

COROLLARY 5.2. *There is a median algorithm for n elements which requires at most $5n + O(n)$ comparisons.*

6. HYPERPAIRS

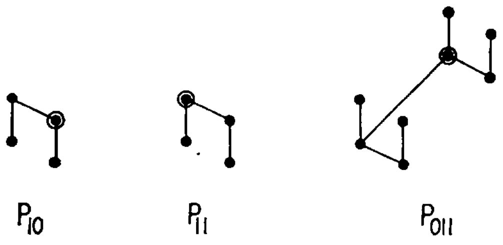
The proof of Theorem 3.1 led us to the partial orders we define here as "hyperpairs," which are an important product of the factories we shall describe.

A *hyperpair* P_w, where w is a binary string, is a finite partial order with a distinguished element, the *center*, defined recursively by

(i) P_λ (λ is the empty string) is a single element, and

(ii) P_{w1} or P_{w0} is obtained from two disjoint copies of P_w by comparing the centers and taking the higher or lower of these respectively as the new center.

Examples of hyperpairs, with the centers circled, are



THEOREM 6.1. *Suppose* $|w| = m$ *and* P_w *has center* c.

(i) P_w *has* 2^m *elements, and exactly* $2^m - 1$ *comparisons are required to produce it.*

(ii) *If* w *has* h *zeros and* $m - h$ *ones then* c *together with those elements greater than (less than)* c *in the partial order are* 2^h *elements* (2^{m-h} *elements*) *forming a* P_{0^h} ($P_{1^{m-h}}$)

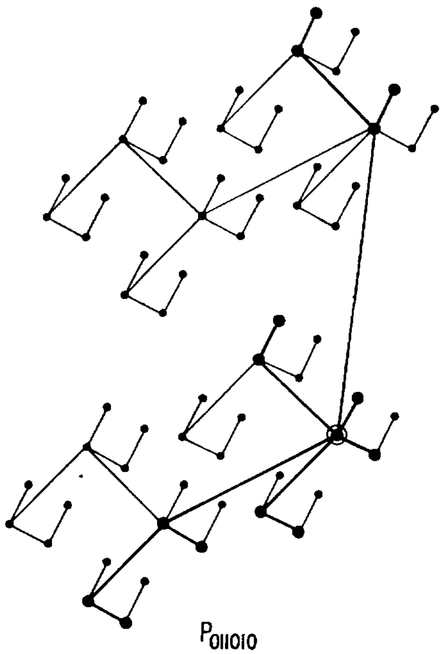


FIGURE 3

with center c. The elements greater than c form a disjoint set of hyperpairs P_λ, P_0, $P_{00}, \ldots, P_{0^{h-1}}$, *and the elements less than c form a disjoint set of hyperpairs* P_λ, P_1, $P_{11}, \ldots, P_{1^{m-h-1}}$

(iii) *If w has h zeroes and h ones then* P_w *contains* $S_k{}^k$ *with centre c for any* $k \leqslant 2^h - 1$.

Proof. (i) and (ii) may be proved by induction on m. (iii) follows from (ii) ▌

This theorem is illustrated by Fig. 3, in which $w = 011010$, $h = 3$, and $k = 7$. The elements of the $S_7{}^7$ contained in the hyperpair are drawn boldly.

7. Accounting Principles

We have been representing partial orders by their Hasse diagrams.

It happens that the partial orders which we need to describe are representable by acyclic diagrams. Each comparison performed combines two acyclic components into one. For ease of description we sometimes ignore parts of the partial order by removing edges from the diagram.

To assess the total number of comparisons used by an algorithm we have found it convenient to count the number of edges, r, removed in this way. Initially there are no edges, and finally, if we have preserved acyclicity, there will be $n - 1$, thus the number of comparisons made must be $n - 1 + r$. The suitability of this convention depends on the special nature of our algorithms.

8. Pruning

In Theorem 6.1 (iii) we showed that suitable hyperpairs contain $S_k{}^k$. In order to separate the subgraph containing just these $2k + 1$ elements from the rest, some of the edges must be broken. In Fig. 3 we find that 15 edges need to be removed. This operation is called *pruning*.

Let $pr_1(w)$, the *upward-pruning cost for* P_w, be the number of edges that must be broken to remove all elements except the center and those greater than the center. The *downward-pruning cost for* P_w, $pr_0(w)$ is defined similarly. In Fig. 3 we can count $pr_1(011010) = 10$ and $pr_0(011010) = 11$.

The special hyperpairs H_r *for* $r \geqslant 0$ are defined by:

$$\begin{aligned} H_0 &= P_\lambda, \\ H_1 &= P_0, \\ H_2 &= P_{01}, \end{aligned}$$

and

$$H_{2t} = P_{01(10)^{t-1}},$$
$$H_{2t+1} = P_{01(10)^{t-1}}, \qquad \text{for} \quad t \geqslant 1.$$

Thus $| H_r | = 2^r$. Note the curious initial irregularity of the binary sequences used. The effect of this will be to equalize upward- and downward-pruning costs.

LEMMA 1. (i) $pr_1(011) = 2, pr_0(01) = 2$;

(ii) *for all* w, $pr_1(w01) = 2pr_1(w) + 1$ *and* $pr_0(w10) = 2pr_0(w) + 1$;

(iii) *for all* $t \geqslant 0$, $pr_1(011(01)^t) = 3 \cdot 2^t - 1$ *and* $pr_0(01(10)^t) = 3 \cdot 2^t - 1$.

Proof. (i) By inspection.

(ii) In Fig. 4 we show a P_{w01} parsed into four connected copies of P_w. It is evident that to upward-prune P_{w01} we must upward-prune two of these copies and remove one further edge. The result for downward-pruning of P_{w10} is by a symmetric argument.

(iii) This follows from (i) and (ii) by induction on t. ▌

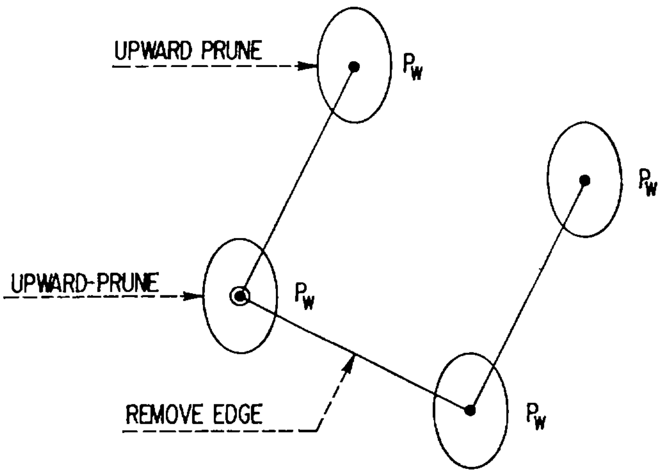


FIGURE 4

Some useful properties of the H_r structures are collected for later reference as:

THEOREM 8.1. (i) *If any edge of an* H_r *is removed, the component not containing the center is an* H_s *for some* $s < r$.

(ii) *For $h \geqslant 1$, H_{2h} can be parsed into its center C and a disjoint set $\{H_0, H_1, H_2, ..., H_{2h-1}\}$ where the centers of $H_0, H_3, H_5, H_7, ..., H_{2h-1}$ are above C and the centers of $H_1, H_2, H_4, H_6, ..., H_{2h-2}$ are below C.*

(iii) *When H_r is upward-pruned for $r = 0, 3, 5, 7, ...$ or downward-pruned for $r = 1, 2, 4, 6, ...$ there remain $2^{\lfloor r/2 \rfloor}$ elements, and the number of edges broken is at most $\frac{3}{2} \cdot 2^{\lfloor r/2 \rfloor}$.*

Proof. (i) and (ii) are proved by induction. (iii) follows from Lemma 1 (iii) and Theorem 6.1 (ii). ▮

COROLLARY 8.2. *If $k \leqslant 2^h - 1$, H_{2h} can be pruned to a subgraph with $2k + 1$ elements containing S_k^k. The pruning cost is at most $3k + 2h$ and the components detached are of the form H_s, $s < 2h$.*

Proof. Let k be expressed in binary, $k = k_0 2^0 + k_1 2^1 + \cdots k_{h-1} 2^{h-1}$, and let H_{2h} be parsed as in (ii). For each i such that $k_i = 1$, upward-prune the term H_j such that $i = \lfloor j/2 \rfloor$ in the sequence $H_0, H_3, H_5, H_7, ..., H_{2h-1}$. By (iii), this yields k elements above C and breaks at most $\frac{3}{2}k$ edges. For each i such that $k_i = 0$, detach the corresponding H_j above C. This breaks at most h edges. Similarly, if $k_i = 1$, downward-prune the term H_j such that $i = \lfloor j/2 \rfloor$ in the sequence $H_1, H_2, H_4, H_6, ..., H_{2h-2}$. By (iii), this yields k elements below C and breaks at most $\frac{3}{2}k$ edges. If $k_i = 0$, detach the corresponding H_j below C. This breaks at most h edges. This procedure yields S_k^k and breaks at most $3k + 2h$ edges. ▮

9. SIMPLE FACTORIES FOR S_k^k

We describe a factory F_k which produces partial orders with $2k + 1$ elements containing S_k^k. Any such partial order will be denoted by $\bar{S}_k^k$. Let h satisfy $2^{h-1} \leqslant k \leqslant 2^h - 1$. F_k uses the following pair of interrelated processes.

Hyperpairing. Whenever there is a pair of H_r's in the factory for some $r < 2h$, their centers are compared to produce a new H_{r+1}. Whenever an H_{2h} is produced it is sent to the pruning process.

Pruning. Each H_{2h} is pruned to an $\bar{S}_k^k$ as described in Corollary 8.2. The detached hyperpairs H_s with $s < 2h$ are returned to the hyperpairing process. The $\bar{S}_k^k$ is output from the factory.

When F_k comes to a halt there can be at most one copy of H_r for each $r < 2h$ in the factory and so

$$R_k = 2^{2h} - 1 < 4k^2$$

To produce m outputs of $\bar{S}_k^k$, at most $(3k + 2h)m$ edges are broken during pruning, $2km$ edges are output in the $\bar{S}_k^k$'s and at most $4k^2$ edges remain in the residual graphs in the factory. Thus the total number of comparisons is at most

$$(5k + 2h)m + 4k^2$$

and so $U_k \lesssim 5k$ and $I_k \leqslant 4k^2$.

This specification of F_k proves Theorem 4.1 as promised.

10. Improvements

The first, and most obvious, improvement to the basic algorithm of Section 5 arises from the observation that the elements returned from the top or bottom halves of the $\bar{S}_k^k$'s eliminated at the ends of the chain can be broken into $(k - 1)/2$ pairs and one singleton. Whenever a pair, instead of two singletons, is input to the hyperpairing process of F_k, one comparison is saved. The total number of comparisons thus saved in the basic algorithm is $\frac{1}{2}n + o(n)$, which yields a median algorithm with only $4.5n + o(n)$ comparisons.

A second more fundamental improvement comes by the new process of "grafting." When an H_{2h} has been completed by the hyperpairing process, instead of pruning it immediately, we compare new singleton elements with its center, at the cost of only one per element, until k of the new elements are all above or all below the center. The new elements may be used in the final $\bar{S}_k^k$ allowing more of the hyperpairs attached to the center to be simply detached, breaking just one edge, instead of being upward- or downward-pruned which breaks many edges. For simplicity we can drop the first improvement described above so that the factory input consists just of singletons. The number of edges broken in the construction of each $\bar{S}_k^k$ using grafting is at most $\frac{3}{2}k + 2h$ (see Theorem 8.1) instead of approximately $3k$. This saving allows a median algorithm using at most $3.5n + o(n)$ comparisons, though in this case the more significant result is the following:

THEOREM 10.1. *For any k, there is a factory F_k for S_k^k with $U_k \sim 3.5k$.*

This is our best upper bound on the unit cost when producing large numbers of S_k^k's. It coincides remarkably with the *lower* bound proved in [4] for the production of a *single* S_k^k.

Our final task is to combine the ideas of the two improvements described above and to give the resulting algorithm in greater detail.

11. The Final Algorithm

We find it natural to describe the algorithm in terms of interconnecting processes with "pipelines" conveying partially ordered sets from the output of one process to the input of another. There are also two reservoirs, one containing singletons and the other containing pairs. A flow-diagram illustrating the interconnections is given in Fig. 5.

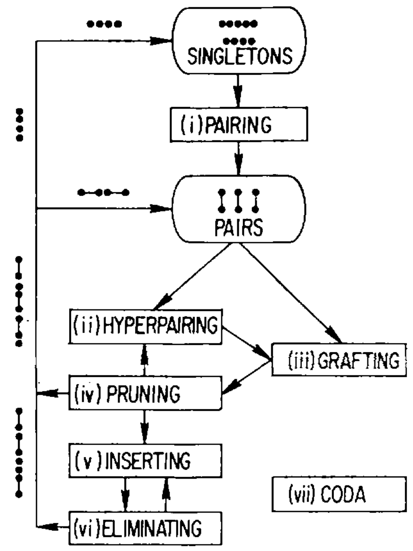


Figure 5

Initially all n elements are in the singleton reservoir.

(i) *Pairing.* Singletons are input from the singleton reservoir whenever possible, compared in disjoint pairs and output to the pair reservoir.

(ii) *Hyperpairing.* If no other process is active, this process attempts to produce one new H_{2h} as described in Section 9. If necessary a pair may be input from the pair reservoir. Any new H_{2h} is output to the grafting process.

(iii) *Grafting.* This process is applied to each H_{2h} from (ii). It uses a supply of pairs input from the pair reservoir. At any stage of the grafting of an H_{2h} the center has edges to a number of singletons and pairs as shown in Fig. 6.

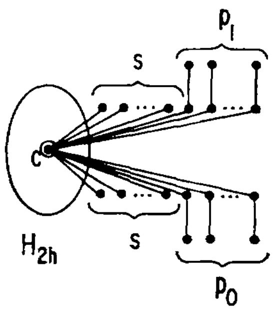
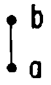


FIGURE 6

Initially, $s = p_1 = p_0 = 0$. As long as $s + 2\max\{p_1, p_0\} \leqslant k - 2$, a new pair



is compared with the center c in the following way:

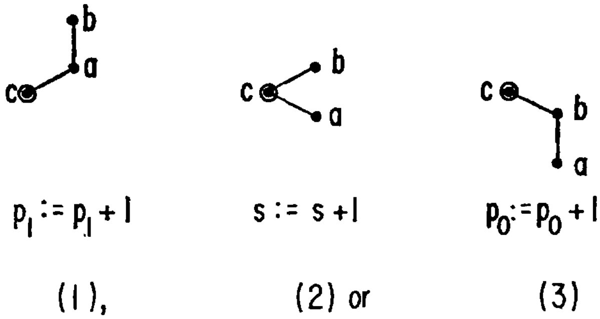
If $p_1 \geqslant p_0$ then

(A) compare c with a, and if $c > a$ compare c with b,

otherwise

(B) compare c with b, and if $c < b$ compare c with a.

The result of these comparisons is



In Case (2), the edge ab has been broken and s has been increased by one. In case (B1) the edge cb, and in Case (A3) the edge ca, has been broken. The number of edges broken up to a given stage is at most $s + \min\{p_1, p_0\} + \min\{p_1 + 1, p_0\}$.

When $s + 2\max\{p_1, p_0\} = k$ or $k - 1$, the resulting partial order is passed to the pruning process.

(iv) *Pruning.* When an input is received from (iii) it is to be pruned to an $\bar{S}_k^k$ in such a way that as many as possible of the hyperpairs attached to the center are detached intact. Thus $k - (s + 2p_1)$ elements need to be produced by upward-pruning from the upper half of the H_{2h}, and $k - (s + 2p_0)$ by downward-pruning from the lower half. From Theorem 8.1 (ii) and (iii) we see that, by pruning an appropriate subset of the sub-hyperpairs of which H_{2h} is composed, any such numbers of elements may be obtained precisely. By Theorem 8.1 (iii) the total of upward- and downward-pruning costs is bounded above by $\frac{3}{2} \times$ (total number of elements produced). In addition at most $2h$ edges are broken to detach complete sub-hyperpairs.

The number of edges broken during grafting and pruning varies with different $\bar{S}_k^k$'s. The accounting is made easier if we also assess at this point the number of edges which may be broken when and if such an $\bar{S}_k^k$ is ultimately destroyed at one end of the chain. Either the upper or lower half must be decomposed into pairs and singletons for recycling to the reservoirs. The number of edges broken consists of one to detach each pair and one to detach each singleton, and this is bounded above by $\frac{1}{2}(k - s - 1) + s + 1 = \frac{1}{2}(k + s + 1)$, since there will be at most $s + 1$ singletons in the half of this $\bar{S}_k^k$ which is recycled.

The total of edges broken during grafting, pruning and the final destruction of the $\bar{S}_k^k$ is at most

$$\begin{aligned} &s + 2\min\{p_1, p_0\} + 1 + \tfrac{3}{2} \cdot (2k - 2s - 2p_1 - 2p_0) + 2h + \tfrac{1}{2}(k + s + 1) \\ &\qquad \leqslant \tfrac{7}{2}k + 2h + \tfrac{3}{2} - \tfrac{3}{2} \cdot (s + 2\max\{p_1, p_0\}) \leqslant 2k + 2h + 3 \end{aligned}$$

since $s + 2\max\{p_1, p_0\} \geqslant k - 1$.

(v) *Inserting.* This process receives the new partial order $\bar{S}_k^k$ from (iv) and inserts its center into the totally ordered chain of centers of previous partial orders, using a binary insertion algorithm. The number of edges broken during insertion is one less than the number of comparisons made and so is at most $\log_2(n/k)$ since there are never more than $n/(2k + 1)$ centers in the chain.

(vi) *Eliminating.* This process is invoked only when none of the previous processes can continue, therefore we can assume that these processes are retaining at most the following elements:

in (i) a singleton,
in (ii) one each of $H_1, H_2, \ldots, H_{2h-1}$,
in (iii) one H_{2h} and $2k - 4$ other elements,
in (iv) none,

and both reservoirs are empty. This makes a total of at most $2^{2h+1} - 1 + 2k - 4 \leqslant 8k^2 + 2k - 5$ since $2^{h-1} \leqslant k$.

Suppose the chain of centers maintained by process (v) is of length t at this time. Then, as argued in the proof of Theorem 5.1, if $t - 1 > 8k^2 + 2k - 5$ the center of the highest $\bar{S}_k^k$ in the chain must be above the median. Therefore this center and all elements in the upper half of this $\bar{S}_k^k$ may be eliminated from the algorithm. Similarly the center and lower half of the lowest $\bar{S}_k^k$ in the chain may be eliminated. The edges connecting the eliminated elements to the main component will be counted later.

The lower half of the highest $\bar{S}_k^k$ and the upper half of the lowest $\bar{S}_k^k$ are broken down into pairs and singletons and recycled to the respective reservoirs. The edges broken in the course of this have been counted already in (iv).

(vii) *Coda.* This finishing process is invoked when no other process can continue, so at most $8k^2 + 2k - 5$ elements are retained by processes (i)–(iv), and the chain maintained by (v) and (vi) is of length $t \leqslant 1 + 8k^2 + 2k - 5$. The total number of elements remaining is therefore $O(k^3)$. The median of the original set is the median of this remaining set, which may be found using $O(k^3)$ comparisons.

12. Final Account

The number of edges broken during the construction, insertion, and destruction of one $\bar{S}_k^k$, plus the number of edges connecting the $k + 1$ elements which are eliminated is at most

$$2k + 2h + 3 + \log_2(n/k) + k + 1,$$

which gives a number of $3 + O((\log n)/k)$ *per element eliminated.* Since $n - O(k^3)$ elements are eliminated in this way and the coda uses $O(k^3)$ further comparisons, the total for the whole algorithm is at most

$$3n + O((n \log n)/k + k^3)$$

Choosing $k \sim (n \log_2 n)^{1/4}$ we reach our final upper bound of

$$3n + O((n \log n)^{3/4}) \sim 3n.$$

Acknowledgments

We were encouraged and supported in this work by useful conversations with Michael Rabin and several members of the Theory of Computation Group at Project MAC.

REFERENCES

1. M. Blum, R. Floyd, V. Pratt, R. Rivest, and R. Tarjan, Time bounds for selection, *J. Comput. Systems Sci.* **7** (1973), 448–461.
2. D. Kirkpatrick, Topics in the complexity of combinatorial algorithms, Tech. Report No. 74 (Dec. 1974), Dept. of Comp. Sci., Univ. of Toronto.
3. D. Knuth, "The Art of Computer Programming," Vol. 3. Addison–Wesley, Reading, Mass., 1973.
4. V. Pratt and F. Yao, On lower bounds for computing the ith largest element, *in* "Proc. 14th Ann. IEEE Symp. on Switching and Automata Theory," Iowa City, Iowa, pp. 70–81, 1973.
5. F. Yao, On lower bounds for selection problems, MAC TR–121, Project MAC, Mass. Inst. of Technology.