ON LOWER BOUNDS FOR SELECTING THE MEDIAN

DORIT DOR*, JOHAN HSTAD†, STAFFAN ULFBERG†, AND URI ZWICK*

Abstract. We present a reformulation of the $2n + o(n)$ lower bound of Bent and John for the number of comparisons needed for selecting the median of n elements. Our reformulation uses a weight function. Apart from giving a more intuitive proof for the lower bound, the new formulation opens up possibilities for improving it. We use the new formulation to show that any *pair-forming* median finding algorithm, i.e., a median finding algorithm that starts by comparing $\lfloor n/2 \rfloor$ disjoint pairs of elements, must perform, in the worst case, at least $2.01n + o(n)$ comparisons. This provides strong evidence that selecting the median requires at least $cn + o(n)$ comparisons, for some $c > 2$.

Key words. median selection, comparison algorithms, lower bounds

AMS subject classifications. 68Q25, 68R05, 06A07

1. Introduction.

Sorting and selection problems have received extensive attention by computer scientists and mathematicians for a long time. Comparison based algorithms for solving these problems work by performing pairwise comparisons between the elements until the relative order of all elements is known, in the case of sorting, or until the i-th largest element among the n input elements is found, in the case of selection.

Sorting in a comparison based computational model is quite well understood. Any deterministic algorithm can be modeled by a decision tree in which all internal nodes represent a comparison between two elements; every leaf represents a result of the computation. Since there must be at least as many leaves in the decision tree as there are possible re-orderings of n elements, all algorithms that sort n elements use at least $\lceil \log n! \rceil \geq n \log n - n \log e + o(n) \approx n \log n - 1.44n + o(n)$ comparisons in the worst case. (All logarithms in this paper are base 2 logarithms.) The best known sorting method, called *merge insertion* by Knuth [9], is due to Lester Ford Jr. and Selmer Johnson [7]. It sorts n elements using at most $n \log n - 1.33n + o(n)$ comparisons. Thus, the gap between the upper and lower bounds is very narrow in that the error in the second order term is bounded by $0.11n$.

The problem of finding the median is the special case of selecting the i-th largest in an ordered set of n elements, when $i = \lceil n/2 \rceil$. Although much effort has been put into finding the exact number of required comparisons, there is still an annoying gap between the best upper and lower bounds currently known.

Knowing how to sort, we could select the median by first sorting, and then selecting the middle-most element; it is quite evident that we could do better, but how much better? This question received a somewhat surprising answer when Blum et al. [3] showed, in 1973, how to determine the median in linear time using at most $5.43n$ comparisons. This result was improved upon in 1976 when Schnhage, Paterson, and Pippinger [13] presented an algorithm that uses only $3n + o(n)$ comparisons. Their

*School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel. *E-mail:* `dorit@checkpoint.com` and `zwick@post.tau.ac.il`.

†Department of Numerical Analysis and Computing Science, Royal Institute of Technology, 100 44 Stockholm, Sweden. *E-mail:* {`johanh,staffanu`}`@nada.kth.se`.

main invention was the use of *factories* which mass-produce certain partial orders that can be easily merged with each other.

This remained the best algorithm for almost 20 years, until Dor and Zwick [5] pushed down the number of comparisons a little bit further to $2.95n + o(n)$ by adding *green factories* that recycle debris from the merging process used in the algorithm of [13].

The first non-trivial lower bound for the problem was also presented, in 1973, by Blum et al. [3] using an adversary argument. Their $1.5n$ lower bound was subsequently improved to $1.75n + o(n)$ by Pratt and Yao [12] in 1973. Then Yap [14], and later Munro and Poblete [10], improved it to $\frac{38}{21}n + O(1)$ and $\frac{79}{43}n + O(1)$, respectively. The proofs of these last two bounds are long and complicated.

In 1979, Fussenegger and Gabow [8] proved a $1.5n + o(n)$ lower bound for the median using a new proof technique. Bent and John [2] used the same basic ideas when they gave, in 1985, a short proof that improved the lower bound to $2n + o(n)$, which is currently the best available. Thus, the uncertainty in the coefficient of n is larger for finding the median than it is for sorting, even though the linear term is the second order term in the case of sorting.

Since our methods are based on the proof by Bent and John, let us describe it in some detail. Given the decision tree of a comparison based algorithm, they invented a method to prune it that yields a collection of pruned trees. Then, lower bounds for the number of pruned trees and for their number of leaves are obtained. A final argument saying that the leaves of the pruned trees are almost disjoint then gives a lower bound for the size of the decision tree.

In Section 2 we reformulate the proof by Bent and John by assigning weights to each node in the decision tree. The weight of a node v corresponds to the total number of leaves in subtrees with root v in all pruned trees where v occurs in the proof by Bent and John. The weight of the root is approximately 2^{2n}; we show that every node v in the decision tree has a child whose weight is at least half the weight of v, and that the weights of all the leaves are small.

When the proof is formulated in this way, it becomes more transparent, and one can more easily study individual comparisons, to rule out some as being bad from the algorithm's point of view.

For many problems, such as finding the maximal or the minimal element of an ordered set, and finding the maximal *and* minimal element of an ordered set, there are optimal algorithms that start by making $\lfloor n/2 \rfloor$ pairwise comparisons between singleton elements. We refer to algorithms that start in this way as being *pair-forming*. It has been discussed whether there are optimal pair-forming algorithms for all partial orders, and in particular this question was posed as an open problem by Aigner [1]. Some examples were then found by Chen [4], showing that pair-forming algorithms are not always optimal.

It is interesting to note that the algorithms in [5] and [13] are both pair-forming. It is still an open problem whether there are optimal pair-forming algorithms for finding the median.

In Section 3 we use our new approach to prove that any pair-forming algorithm uses at least $2.01227n + o(n)$ comparisons to find the median.

Dor and Zwick [6] have recently been able to extend the ideas described here and obtain a $(2+\epsilon)n$ lower bound, for some tiny $\epsilon > 0$, on the number of comparisons performed, in the worst case, by any median selection algorithm.

2. Bent and John revisited. Bent and John [2] proved that $2n + o(n)$ comparisons are required for selecting the median. Their result, in fact, is more general and provides a lower bound for the number of comparisons required for selecting the i-th largest element, for any $1 \leq i \leq n$. We concentrate here on median selection although our results, like those of Bent and John, can be extended to general i.

Although the proof given by Bent and John is relatively short and simple, we here present a reformulation. There are two reasons for this: the first is that the proof gets more transparent; the second is that this formulation makes it easier to study the effect of individual comparisons.

THEOREM 2.1 (Bent and John [2]). *Finding the median requires $2n + o(n)$ comparisons.*

Proof.

Any deterministic algorithm for finding the median can be represented by a decision tree T, in which each internal node v is labeled by a comparison $a : b$. The two children of such a node, $v_{a<b}$ and $v_{a>b}$, represent the outcomes $a < b$ and $a > b$, respectively. We assume that decision trees do not contain redundant comparisions between elements whose relative order has already been established.

We consider a universe U containing n elements. For every node v in T and subset C of U we make the following definitions:

$$\max_v(C) = \left\{ a \in C \;\middle|\; \begin{array}{l} \text{every comparison } a : b \text{ above } v \\ \text{with } b \in C \text{ had outcome } a > b \end{array} \right\},$$

$$\min_v(C) = \left\{ a \in C \;\middle|\; \begin{array}{l} \text{every comparison } a : b \text{ above } v \\ \text{with } b \in C \text{ had outcome } a < b \end{array} \right\}.$$

Before we proceed with the proof that selecting the median requires $2n + o(n)$ comparisons, we present a proof of a somewhat weaker result. We assume that U contains $n = 2m$ elements and show that selecting the two middlemost elements requires $2n + o(n)$ comparisons. The proof in this case is slightly simpler, yet it demonstrates the main ideas used in the proof of the theorem.

We define a weight function on the nodes of T. This weight function satisfies the following three properties: (i) the weight of the root is $2^{2n+o(n)}$. (ii) each internal node v has a child whose weight is at least half the weight of v. (iii) the weight of each leaf is small.

For every node v in the decision tree, we keep track of subsets A of size m which may contain the m largest elements with respect to the comparisons already made. Let $\mathcal{A}(v)$ contain all such sets which are called *upper half compatible* with v. The As are assigned weights which estimate how far from a solution the algorithm is, assuming that the elements in A are the m largest. The weight of every $A \in \mathcal{A}(v)$ is defined as

$$w_v^1(A) = 2^{|\min_v(A)| + |\max_v(\bar{A})|},$$

and the weight of a node v is defined as

$$w(v) = \sum_{A \in \mathcal{A}(v)} w_v^1(A).$$

The superscript 1 in $w_v^1(A)$ is used as we shall shortly have to define a second weight function $w_v^2(B)$.

case		$w^1_{v_{a<b}}(A)$	$w^1_{v_{a>b}}(A)$
$a \in A$	$b \in A$	$\frac{1}{2}$ or 1	$\frac{1}{2}$ or 1
$a \in A$	$b \in \bar{A}$	0	1
$a \in \bar{A}$	$b \in A$	1	0
$a \in \bar{A}$	$b \in \bar{A}$	$\frac{1}{2}$ or 1	$\frac{1}{2}$ or 1

TABLE 2.1
The weight of a set $A \in \mathcal{A}(v)$ in the children of a node v, relative to its weight in v.

In the root r of T, all subsets of size m of U are upper half compatible with r so that $|\mathcal{A}(r)| = \binom{2m}{m}$. Also, each $A \in \mathcal{A}(r)$ has weight 2^{2m}, and we find, as promised, that

$$w(r) = 2^{2m}\binom{2m}{m} = 2^{2n+o(n)}.$$

Consider the weight $w^1_v(A)$ of a set $A \in \mathcal{A}(v)$ at a node v labeled by the comparison $a : b$. What are the weights of A in v's children? This depends on which of the elements a and b belongs to A (and on which of them is minimal in A or maximal in $\bar{A}$). The four possible cases are considered in Table 2.1. The weights given there are relative to the weight $w^1_v(A)$ of A at v. A zero indicates that A is no longer compatible with this child and thus does not contribute to its weight. The weight $w^1_{v_{a<b}}(A)$, when $a, b \in A$, for example, is $\frac{1}{2}w^1_v(A)$, if $b \in \min_v(A)$, and is $w^1_v(A)$, otherwise. As can be seen, v always has at least one child in which the weight of A is at least half its weight at v. Furthermore, in each one of the four cases, $w^1_{v_{a<b}}(A) + w^1_{v_{a>b}}(A) \geq w^1_v(A)$.

Each leaf v of the decision tree corresponds to a state of the algorithm in which the two middlemost elements were found. There is therefore only one set A left in $\mathcal{A}(v)$. Since we have identified the minimum element in A and the maximum element in $\bar{A}$, we get that $w^1_v(A) = 4$. So, if we follow a path from the root of the tree and repeatedly descend to the child with the largest weight, we will, when we eventually reach a leaf, have performed at least $2n + o(n)$ comparisons.

We now prove that selecting the median also requires at least $2n + o(n)$ comparisons. To make the median well defined we assume that $n = 2m - 1$. The problem that arises in the above argument is that the weights of the leaves in T, when the selection of the median, and not the two middlemost elements, is considered, are not necessarily small enough: it is possible to know the median without knowing any relations between elements in $\bar{A}$ (which now contains $m - 1$ elements); this is remedied as follows.

In a node v where the algorithm is close to determining the minimum element in A, we essentially force it to determine the largest element in $\bar{A}$ instead. This is done by moving an element a_0 out of A and creating a set $B = \bar{A} \cup \{a_0\}$. This set is *lower half compatible* with v and the median is the maximum element in B. By a suitable choice of a_0, most of $\max_v(\bar{A})$ is in $\max_v(B)$. A set B is lower half compatible with v if $|B| = m$ and it may contain the m smallest elements in U. We keep track of Bs in the *multiset* $\mathcal{B}(v)$.

For the root r of T, we let $\mathcal{A}(r)$ contain all subsets of size m of U as before, and let $\mathcal{B}(r)$ be empty. We exchange some As for Bs as the algorithm proceeds. The

case		$w^2_{v_{a<b}}(B)$	$w^2_{v_{a>b}}(B)$
$a \in B$	$b \in B$	$\frac{1}{2}$ or 1	$\frac{1}{2}$ or 1
$a \in B$	$b \in \bar{B}$	1	0
$a \in \bar{B}$	$b \in B$	0	1
$a \in \bar{B}$	$b \in \bar{B}$	1	1

TABLE 2.2
The weight of a set $B \in \mathcal{B}(v)$ in the children of a node v, relative to its weight in v.

weight of a set B is defined as

$$w^2_v(B) = 2^{|\max_v(B)|}.$$

The weight of B estimates how far the algorithm is from a solution, assuming that the elements in B are the m smallest elements. The weight of a node v is now defined to be

$$w(v) = \sum_{A \in \mathcal{A}(v)} w^1_v(A) + 2^{4\sqrt{n}} \sum_{B \in \mathcal{B}(v)} w^2_v(B).$$

In the beginning of an algorithm (in the upper part of the decision tree), the weight of a node is still the sum of the weights of all As, and therefore $w(r) = 2^{2n+o(n)}$.

We now define $\mathcal{A}(v)$ and $\mathcal{B}(v)$ for the rest of T more exactly. For any node v in T, except the root, simply copy $\mathcal{A}(v)$ and $\mathcal{B}(v)$ from the parent node and remove all sets that are not upper or lower half compatible with v, respectively. We ensure that the weight of every leaf is small by doing the following: If, for some $A \in \mathcal{A}(v)$ we have $|\min_v(A)| = \lceil 2\sqrt{n}\rceil$, we select an element $a_0 \in \min_v(A)$ which has been compared to the fewest number of elements in $\bar{A}$; we then remove the set A from $\mathcal{A}(v)$ and add the set $B = \bar{A} \cup \{a_0\}$ to $\mathcal{B}(v)$.

Note that at the root, $|\min_r(A)| = m$ for all $A \in \mathcal{A}(r)$, and that this quantity decreases by at most one for each comparison until a leaf is reached. In a leaf v the median is known; thus, $\mathcal{A}(v)$ is empty.

LEMMA 2.2. *Let $\mathcal{A}(v)$ and $\mathcal{B}(v)$ be defined by the rules described above. Then, every internal node v (labeled $a : b$) in T has a child with at least half the weight of v, i.e., $w(v_{a<b}) \geq w(v)/2$ or $w(v_{a>b}) \geq w(v)/2$.*

Proof. Table 2.1 gives the weights of a set $A \in \mathcal{A}(v)$ at v's children, relative to the weight $w^1_v(A)$ of A at v. Similarly, Table 2.2 gives the weights of a set $B \in \mathcal{B}(v)$ in v's children, relative to the weight $w^2_v(v)$ of B at v. As $w^1_{v_{a<b}}(A) + w^1_{v_{a>b}}(A) \geq w^1_v(A)$ and $w^2_{v_{a<b}}(B) + w^2_{v_{a>b}}(B) \geq w^2_v(B)$, for every $A \in \mathcal{A}(v)$ and $B \in \mathcal{B}(v)$, all that remains to be checked is that the weight does not decrease when a lower half compatible set B replaces an upper half compatible set A. This is covered by Lemma 2.3. ☐

LEMMA 2.3. *If A is removed from $\mathcal{A}(v)$ and B is added in its place to $\mathcal{B}(v)$, and if fewer than $4n$ comparisons have been performed on the path from the root to v, then $2^{4\sqrt{n}} w^2_v(B) > w^1_v(A)$.*

Proof. A set $A \in \mathcal{A}(v)$ is replaced by a set $B = \bar{A} \cup \{a_0\} \in \mathcal{B}(v)$ only when $|\min_v(A)| = \lceil 2\sqrt{n}\rceil$. The element a_0, in such a case, is an element of $\min_v(A)$ that has been compared to the fewest number of elements in $\bar{A}$. If a_0 was compared to at least $2\sqrt{n}$ elements in $\bar{A}$, we get that each element of $\min_v(A)$ was compared to at least $2\sqrt{n}$ elements in $\bar{A}$, and at least $4n$ comparisons have been performed on the path from the root to v, a contradiction. We get therefore that a_0 was compared to fewer than $2\sqrt{n}$ elements of $\bar{A}$ and thus $|\max_v(B)| > |\max_v(\bar{A})| - 2\sqrt{n}$. As a consequence, we get that $4\sqrt{n} + |\max_v(B)| > |\min_v(A)| + |\max_v(\bar{A})|$ and thus $2^{4\sqrt{n}} w_v^2(B) > w_v^1(A)$, as required. □

We now know that the weight of the root is large, and that the weight does not decrease too fast; what remains to be shown is that the weights of the leaves are relatively small. This is established in the following lemma.

Lemma 2.4. *For a leaf v (in which the median is known), $w(v) \le 2m2^{4\sqrt{n}}$.*

Proof. Clearly, the only sets compatible with a leaf of T are the set A containing the m largest elements, and the set B containing the m smallest elements. Since $|\min_v(A)| = |\max_v(B)| = 1$, we get that $w_v^2(B) = 2$ and $A \notin \mathcal{A}(v)$.

Since there are exactly m elements that can be removed from B to obtain a corresponding $\bar{A}$, there can be at most m copies of B in $\mathcal{B}(v)$. □

Let T be a comparison tree that corresponds to a median finding algorithm. If the height of T is at least $4n$, we are done. Otherwise, by starting at the root and repeatedly descending to a child whose weight is at least half the weight of its parent, we trace a path whose length is at least $2n + o(n)$ and Theorem 2.1 follows. □

Let us see how the current formalism gives room for improvement that did not exist in the original proof. The $2n + o(n)$ lower bound is obtained by showing that each node v in a decision tree T that corresponds to a median finding algorithm has a child whose weight is at least half the weight of v. Consider the nodes $v_0, v_1, \ldots, v_\ell$ along the path obtained by starting at the root of T and repeatedly descending to the child with the larger weight, until a leaf is reached. If we could show that sufficiently many nodes on this path have weights strictly larger than half the weights of their parents, we would obtain an improved lower bound for median selection. If $w(v_i) \ge \frac{1}{2}(1+\delta_i)\cdot w(v_{i-1})$, for every $1 \le i \le \ell$, then the length of this path, and therefore the depth of T, is at least $2n + \sum_{i=1}^{\ell} \log_2(1+\delta_i) + o(n)$.

3. An improved lower bound for pair-forming algorithms.

Let v be a node of a comparison tree. An element x is a *singleton* at v if it was not compared above v with any other element. Two elements x and y form a *pair* at v if the elements x and y were compared to each other above v, but neither of them was compared to any other element.

A pair-forming algorithm is an algorithm that starts by constructing $\lfloor n/2 \rfloor = m - 1$ pairs. By concentrating on comparisons that involve elements that are part of pairs, we obtain a better lower bound for pair-forming algorithms.

Theorem 3.1. *A pair-forming algorithm for finding the median must perform, in the worst case, at least* $2.00691n + o(n)$ *comparisons.*

Proof.

It is easy to see that a comparison involving two singletons can be delayed until just before one of them is to be compared for the second time. We can therefore restrict our attention to comparison trees in which the partial order corresponding to each node contains at most two pairs. Allowing only one pair is not enough as algorithms should be allowed to construct two pairs $\{a, b\}$ and $\{a', b'\}$, and then compare an element from $\{a, b\}$ with an element from $\{a', b'\}$.

We focus our attention on nodes in the decision tree in which an element of a pair is compared for the second time and in which the number of non-singletons is at most ϵm, for some $\epsilon < 1$. If v is a node in which the number of non-singletons is at

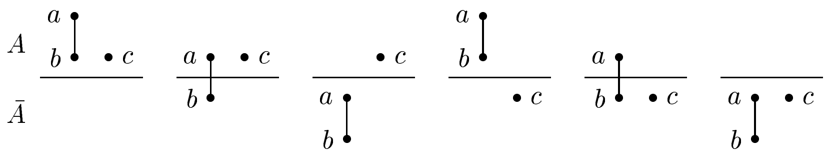

FIG. 3.1. *The six possible ways that a, b, and c may be divided between A and Ā. Note that c is not necessarily a singleton element; it may be part of a larger partial order.*

most ϵm, for some $\epsilon < 1$, then $\mathcal{B}(v)$ is empty and thus $w(v) = \sum_{A\in\mathcal{A}(v)} w_v^1(A)$ and we do not have to consider Table 2.2 for the rest of the section.

Recall that $\mathcal{A}(v)$ denotes the collection of subsets of U size m that are upper half compatible with v. If $H, L \subseteq U$ are subsets of U, of arbitrary size, we let

$$\mathcal{A}_{H/L}(v) = \{A \in \mathcal{A}(v) \mid H \subseteq A \text{ and } L \subseteq \bar{A}\}.$$

We let $w_{H/L}(v)$ be the contribution of the sets of $\mathcal{A}_{H/L}(v)$ to the weight of v, i.e.,

$$w_{H/L}(v) = \sum_{A\in\mathcal{A}_{H/L}(v)} w_v^1(A).$$

For brevity, we write $\mathcal{A}_{h_1\ldots h_r/l_1\ldots l_s}(v)$ for $\mathcal{A}_{\{h_1,\ldots,h_r\}/\{l_1,\ldots,l_s\}}(v)$ and $w_{h_1\ldots h_r/l_1\ldots l_s}(v)$ for $w_{\{h_1,\ldots,h_r\}/\{l_1,\ldots,l_s\}}(v)$.

Before proceeding, we describe the intuition that lies behind the rest of the proof. Consider Table 2.1 from the last section. If, in a node v of the decision tree, the two cases $a \in A$, $b \in \bar{A}$ and $a \in \bar{A}$, $b \in A$ are not equally likely, or more precisely, if the contributions $w_{a/b}(v)$ and $w_{b/a}(v)$ of these two cases to the total weight of v are not equal, there must be at least one child of v whose weight is greater than half the weight of v. The difficulty in improving the lower bound of Bent and John lies therefore at nodes in which the the contributions of the two cases $a \in A$, $b \in \bar{A}$ and $a \in \bar{A}$, $b \in A$ are almost equal. This fact is not so easily seen when looking at the original proof given in [2].

Suppose now that v is a node in which an element a of a pair $\{a, b\}$ is compared with an arbitrary element c and that the number of non-singletons in v is at most ϵm. We assume, without loss of generality, that $a > b$. The weights of a set $A \in \mathcal{A}(v)$ in v's children depend on which of the elements a, b, and c belongs to A, and on whether c is minimal in A or maximal in $\bar{A}$. The six possible ways of dividing the elements a, b, and c between A and $\bar{A}$ are shown in Figure 3.1. The weights of the set A in v's children, relative to the weight $w_v^1(A)$ of A at v, in each one of these six cases are given in Table 3.1. Table 3.1 is similar to Table 2.1 of the previous section, with c playing the role of b. There is one important difference, however. If $a, b, c \in A$, as in the first row of Table 3.1, then the weight of A in $v_{a>c}$ is equal to the weight of A in v. The weight is not halved, as may be the case in the first row of Table 2.1. If the contribution $w_{abc/}(v)$ of the case $a, b, c \in A$ to the weight of v is not negligible, there must again be at least one child of v whose weight is greater than half the weight of v.

The improved lower bound is obtained by showing that if the contributions of the cases $a \in A$, $b \in \bar{A}$ and $a \in \bar{A}$, $b \in A$ are roughly equal, and if most elements in the partial order are singletons, then the contribution of the case $a, b, c \in A$ is non-negligible. The larger the number of singletons in the partial order, the larger is the relative contribution of the weight $w_{abc/}(v)$ to the weight $w(v)$ of v. Thus, whenever

case			$w^1_{v_{a<c}}(A)$	$w^1_{v_{a>c}}(A)$
$a \in A$	$b \in A$	$c \in A$	$\frac{1}{2}$ or 1	1
$a \in A$	$b \in \bar{A}$	$c \in A$	$\frac{1}{2}$ or 1	$\frac{1}{2}$
$a \in \bar{A}$	$b \in \bar{A}$	$c \in A$	1	0
$a \in A$	$b \in A$	$c \in \bar{A}$	0	1
$a \in A$	$b \in \bar{A}$	$c \in \bar{A}$	0	1
$a \in \bar{A}$	$b \in \bar{A}$	$c \in \bar{A}$	$\frac{1}{2}$	$\frac{1}{2}$ or 1

TABLE 3.1
The weight of a set $A \in \mathcal{A}(v)$ in the children of a node v, relative to its weight in v, when the element a of a pair $a > b$ is compared with an arbitrary element c.

an element of a pair is compared for the second time, we make a small gain. The above intuition is made precise in the following lemma:

LEMMA 3.2. *If v is a node in which an element a of a pair $a > b$ is compared with an element c, and if the number of singletons in v is at least $m + 2\sqrt{n}$, then*

$$\begin{aligned} w(v_{a<c}) &\geq \tfrac{1}{2}w(v) + \tfrac{1}{2}(w_{c/a}(v) - w_{a/c}(v)) \;, \\ w(v_{a>c}) &\geq \tfrac{1}{2}w(v) + \tfrac{1}{2}(w_{a/c}(v) - w_{c/a}(v) + w_{abc/}(v)) \;. \end{aligned}$$

Proof. Both inequalities follow easily by considering the entries in Table 3.1. To obtain the second inequality, for example, note that $w(v_{a>c}) \geq \frac{1}{2}(w(v) + w_{abc/}(v) - w_{c/ab}(v) + w_{ab/c}(v) + w_{a/bc}(v))$. As $w_{c/ab}(v) = w_{c/a}(v)$ and $w_{ab/c}(v) + w_{a/bc}(v) = w_{a/c}(v)$, the second inequality follows. ☐

It is worth pointing out that in Table 3.1 and in Lemma 3.2, we only need to assume that $a > b$; we do not use the stronger condition that $a > b$ is a pair. This stronger condition is crucial however in the sequel, especially in Lemma 3.4.

To make use of Lemma 3.2 we need bounds on the relative contributions of the different cases. The following lemma is a useful tool for determining such bounds.

LEMMA 3.3. *Let $G = (V_1, V_2, E)$ be a bipartite graph. Let δ_1 and δ_2 be the minimal degree of the vertices of V_1 and V_2, respectively. Let Δ_1 and Δ_2 be the maximal degree of the vertices of V_1 and V_2, respectively. Assume that a positive weight function w is defined on the vertices of G such that $w(v_1) = r \cdot w(v_2)$, whenever $v_1 \in V_1$, $v_2 \in V_2$ and $(v_1, v_2) \in E$. Let $w(V_1) = \sum_{v_1 \in V_1} w(v_1)$ and $w(V_2) = \sum_{v_2 \in V_2} w(v_2)$. Then,*

$$r\frac{\delta_2}{\Delta_1} \cdot w(V_2) \;\leq\; w(V_1) \;\leq\; r\frac{\Delta_2}{\delta_1} \cdot w(V_2).$$

Proof. Let $v_1(e) \in V_1$ and $v_2(e) \in V_2$ denote the two vertices connected by the edge e. We then have

$$\delta_1 \sum_{v_1 \in V_1} w(v_1) \leq \sum_{e \in E} w(v_1(e)) = r \sum_{e \in E} w(v_2(e)) \leq r\Delta_2 \sum_{v_2 \in V_2} w(v_2).$$

The other inequality follows by exchanging the roles of V_1 and V_2. □

Using Lemma 3.3 we obtain the following basic inequalities.

LEMMA 3.4. *If v is a node in which $a > b$ is a pair and the number of non-singletons in v is at most ϵm, then*

$$\begin{array}{rcccl}
\frac{1}{2}(1-\epsilon)\cdot w_{ac/b}(v) & \le & w_{abc/}(v) & \le & \frac{1}{2(1-\epsilon)}\cdot w_{ac/b}(v) \ , \\
2(1-\epsilon)\cdot w_{c/ab}(v) & \le & w_{ac/b}(v) & \le & \frac{2}{1-\epsilon}\cdot w_{c/ab}(v) \ , \\
\frac{1}{2}(1-\epsilon)\cdot w_{a/bc}(v) & \le & w_{ab/c}(v) & \le & \frac{1}{2(1-\epsilon)}\cdot w_{a/bc}(v) \ , \\
2(1-\epsilon)\cdot w_{/abc}(v) & \le & w_{a/bc}(v) & \le & \frac{2}{1-\epsilon}\cdot w_{/abc}(v) \ .
\end{array}$$

Each one of these inequalities relates a weight, such as $w_{abc/}(v)$, to a weight, such as $w_{ac/b}(v)$, obtained by moving one of the elements of the pair $a > b$ from A to $\bar{A}$. In each inequality we 'lose' a factor of $1-\epsilon$. When the elements a and b are joined together a factor of 2 is introduced. When the elements a and b are separated, a factor of $\frac{1}{2}$ is introduced.

Proof. We present a proof of the inequality $w_{abc/}(v) \le \frac{1}{2(1-\epsilon)}\cdot w_{ac/b}(v)$. The proof of all the other inequalities is almost identical.

Construct a bipartite graph $G = (V_1, V_2, E)$ whose vertex sets are $V_1 = \mathcal{A}_{abc/}(v)$ and $V_2 = \mathcal{A}_{ac/b}(v)$. Define an edge $(A_1, A_2) \in E$ between $A_1 \in \mathcal{A}_{abc/}(v)$ and $A_2 \in \mathcal{A}_{ac/b}(v)$ if and only if there is a singleton $d \in \bar{A}_1$ such that $A_2 = A_1 \setminus \{b\} \cup \{d\}$. Suppose that (A_1, A_2) is such an edge. As $a \notin \min_v(A_1)$ but $a \in \min_v(A_2)$, while all other elements are extremal with respect to A_1 if and only if they are extremal with respect to A_2 (note that $b \in \min_v(A_1)$ and $b \in \max_v(\bar{A}_2)$), we get that $w_v^1(A_1) = \frac{1}{2}\cdot w_v^1(A_2)$.

For every set A of size m, the number of singletons in A is at least $(1-\epsilon)m$ and at most m. We get therefore that the minimal degrees of the vertices of V_1 and V_2 are $\delta_1, \delta_2 \ge (1-\epsilon)m$ and the maximal degrees of V_1 and V_2 are $\Delta_1, \Delta_2 \le m$. The inequality $w_{abc/}(v) \le \frac{1}{2(1-\epsilon)}\cdot w_{ac/b}(v)$ therefore follows from Lemma 3.3. □

Using these basic inequalities we obtain:

LEMMA 3.5. *If v is a node in which $a > b$ is a pair and the number of non-singletons is at most ϵm, for some $\epsilon < 1$, then*

$$\begin{aligned}
w_{abc/}(v) &\ge \tfrac{(1-\epsilon)^2}{(2-\epsilon)^2}\cdot w_{c/}(v) \ , \\
w_{a/c}(v) &\ge \tfrac{(1-\epsilon)(3-\epsilon)}{(2-\epsilon)^2}\cdot w_{/c}(v) \ , \\
w_{c/a}(v) &\le \tfrac{1}{(2-\epsilon)^2}\cdot w_{c/}(v) \ .
\end{aligned}$$

Proof. We present the proof of the first inequality. The proof of the other two inequalities is similar. Using inequalities from Lemma 3.4 we get that

$$\begin{aligned}
w_{c/}(v) &= w_{abc/}(v) + w_{ac/b}(v) + w_{c/ab}(v) \\
&\le w_{abc/}(v) + \tfrac{2}{1-\epsilon}\cdot w_{abc/}(v) + \tfrac{1}{(1-\epsilon)^2}\cdot w_{abc/}(v) \\
&= \tfrac{(2-\epsilon)^2}{(1-\epsilon)^2}\cdot w_{abc/}(v)
\end{aligned}$$

and the first inequality follows. □

We are now ready to show that if v is a node in which an element of a pair is compared for the second time, then v has a child whose weight is greater than half the weight of v. Combining Lemma 3.2 and Lemma 3.5, we get that

$$\begin{aligned}\tfrac{1}{2}\cdot(w(v_{a<c})+w(v_{a>c})) &\geq \tfrac{1}{2}\cdot w(v)+\tfrac{(1-\epsilon)^2}{4(2-\epsilon)^2}\cdot w_{c/}(v)\ ,\\ w(v_{a>c}) &\geq \tfrac{1}{2}\cdot w(v)-\tfrac{\epsilon}{2(2-\epsilon)}\cdot w_{c/}(v)+\tfrac{(1-\epsilon)(3-\epsilon)}{2(2-\epsilon)^2}\cdot w_{/c}(v)\ .\end{aligned}$$

Let $\alpha = w_{c/}(v)/w(v)$ and $1-\alpha = w_{/c}(v)/w(v)$. We get that

$$\begin{aligned}\tfrac{1}{2}\cdot(w(v_{a<c})+w(v_{a>c})) &\geq \left(\tfrac{1}{2}+\tfrac{(1-\epsilon)^2}{4(2-\epsilon)^2}\alpha\right)\cdot w(v)\ ,\\ w(v_{a>c}) &\geq \left(\tfrac{1}{2}-\tfrac{\epsilon}{2(2-\epsilon)}\alpha+\tfrac{(1-\epsilon)(3-\epsilon)}{2(2-\epsilon)^2}(1-\alpha)\right)\cdot w(v)\\ &= \left(\tfrac{1}{2}-\tfrac{3-2\epsilon}{2(2-\epsilon)^2}\alpha+\tfrac{(1-\epsilon)(3-\epsilon)}{2(2-\epsilon)^2}\right)\cdot w(v)\ .\end{aligned}$$

As a consequence, we get that

$$\max\{w(v_{a<c}),w(v_{a>c})\}\geq \max\left\{\tfrac{1}{2}+\tfrac{(1-\epsilon)^2}{4(2-\epsilon)^2}\alpha,\tfrac{1}{2}-\tfrac{3-2\epsilon}{2(2-\epsilon)^2}\alpha+\tfrac{(1-\epsilon)(3-\epsilon)}{2(2-\epsilon)^2}\right\}\cdot w(v)\ .$$

The coefficient of $w(v)$, on the right hand side, is minimized when the two expressions whose maximum is taken are equal. This happens when $\alpha = \frac{2(3-4\epsilon+\epsilon^2)}{7-6\epsilon+\epsilon^2}$. Plugging this value of α into the two expressions, we get that

$$\max\{w(v_{a<c}),w(v_{a>c})\}\geq \tfrac{1}{2}(1+f_1(\epsilon))\cdot w(v)\ ,$$

where

$$f_1(\epsilon)=\frac{(3-\epsilon)(1-\epsilon)^3}{(2-\epsilon)^2(7-6\epsilon+\epsilon^2)}\ .$$

It is easy to check that $f_1(\epsilon)>0$ for $\epsilon<1$.

A *pair-forming* comparison is a comparison in which two singletons are compared to form a pair. A *pair-touching* comparison is a comparison in which an element of a pair is compared for the second time. In a pair-forming algorithm, the number of singletons is decreased only by pair-forming comparisons. Each pair-forming comparison decreases the number of singletons by exactly two. As explained above, pair-forming comparisons can always be delayed so that a pair-forming comparison $a:b$ is immediately followed by a comparison that touches the pair $\{a,b\}$, or by a pair-forming comparison $a':b'$ and then by a comparison that touches both pairs $\{a,b\}$ and $\{a',b'\}$.

Consider again the path traced from the root by repeatedly descending to the child with the larger weight. As a consequence of the above discussion, we get that when the i-th pair-touching comparison along this path is performed, the number of non-singletons in the partial order is at most $4i$. It follows therefore from the remark made at the end of the previous section that the depth of the comparison tree corresponding to any pair-forming algorithm is at least

$$\begin{aligned}&2n+\sum_{i=1}^{m/4}\log_2(1+f_1(\frac{4i}{m}))+o(n)\\ &=2n+\frac{n}{8}\cdot\int_0^1\log_2(1+f_1(t))dt+o(n)\ \approx\ 2.00691n+o(n)\ .\end{aligned}$$

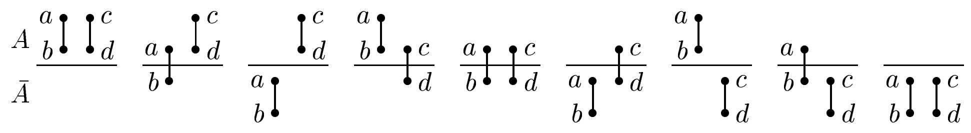

FIG. 3.2. *The nine possible ways that a, b, c, and d may be divided between A and Ā.*

case	$w^1_{v_{a<c}}(A)$	$w^1_{v_{a>c}}(A)$	$w^1_{v_{a<d}}(A)$	$w^1_{v_{a>d}}(A)$
$A \in \mathcal{A}_{abcd/}$	1	1	$\frac{1}{2}$	1
$A \in \mathcal{A}_{acd/b}$	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$A \in \mathcal{A}_{cd/ab}$	1	0	1	0
$A \in \mathcal{A}_{abc/d}$	$\frac{1}{2}$	1	0	1
$A \in \mathcal{A}_{ac/bd}$	$\frac{1}{2}$	$\frac{1}{2}$	0	1
$A \in \mathcal{A}_{c/abd}$	1	0	$\frac{1}{2}$	$\frac{1}{2}$
$A \in \mathcal{A}_{ab/cd}$	0	1	0	1
$A \in \mathcal{A}_{a/bcd}$	0	1	0	1
$A \in \mathcal{A}_{/abcd}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1

TABLE 3.2
The weight of a set $A \in \mathcal{A}(v)$ in the children of a node v, relative to its weight in v, when the element a of a pair $a > b$ is compared with an element of a pair $c > d$.

This completes the proof of Theorem 3.1. □

The worst case in the proof above is obtained when the algorithm converts all the elements into *quartets*. A quartet is a partial order obtained by comparing elements contained in two disjoint pairs. In the proof above, we analyzed cases in which an element a of a pair $a > b$ is compared with an arbitrary element c. If the element c is also part of a pair, a tighter analysis is possible. By performing this anaylsis we can improve Theorem 3.1.

THEOREM 3.6. *A pair-forming algorithm for finding the median must perform, in the worst case, at least* $2.01227n + o(n)$ *comparisons.*

Proof. Consider comparisons in which the element from a pair $a > b$ is compared with an element of a pair $c > d$. The nine possible ways of dividing the elements a, b, c, and d among A and $\bar{A}$ are depicted in Figure 3.2. We may assume, without loss of generality, that the element a is compared with either c or with d.

Let v be a node of the comparison tree in which $a > b$ and $c > d$ are pairs and which one of the comparions $a : c$ or $a : d$ is performed. Let $A \in \mathcal{A}(v)$. The weights of a set A in v's children, relative to the weight $w^1_v(A)$ of A at v, in each one of these nine cases are given in Table 3.2. The two possible comparisons $a : c$ and $a : d$ are considered separately. The following equalities are easily verified.

LEMMA 3.7. *If $a > b$ and $c > d$ are pairs in v then*

$$\begin{aligned} w_{acd/b}(v) &= w_{abc/d}(v) \, , \\ w_{cd/ab}(v) &= w_{ab/cd}(v) \, , \end{aligned}$$

$$\begin{aligned} w_{c/abd}(v) &= \ w_{a/bcd}(v) \ , \\ w_{ac/bd}(v) &= 4 \cdot w_{ab/cd}(v) \ . \end{aligned}$$

The following inequalities are analogous to the inequalities of Lemma 3.4.

LEMMA 3.8. *If $a > b$ and $c > d$ are pairs in v and if the number of non-singletons in v is at most ϵm, for some $\epsilon < 1$, then*

$$\begin{array}{rcccl} \frac{1}{2}(1-\epsilon)w_{abc/d}(v) & \leq & w_{abcd/}(v) & \leq & \frac{1}{2(1-\epsilon)}w_{abc/d}(v) \ , \\ 2(1-\epsilon)w_{ab/cd}(v) & \leq & w_{abc/d}(v) & \leq & \frac{2}{1-\epsilon}w_{ab/cd}(v) \ , \\ \frac{1}{2}(1-\epsilon)w_{a/bcd}(v) & \leq & w_{ab/cd}(v) & \leq & \frac{1}{2(1-\epsilon)}w_{a/bcd}(v) \ , \\ 2(1-\epsilon)w_{/abcd}(v) & \leq & w_{a/bcd}(v) & \leq & \frac{2}{1-\epsilon}w_{/abcd}(v) \ . \end{array}$$

Consider first the comparison $a : c$. By examining Table 3.2 and using the equalities of Lemma 3.7, we get that

$$\begin{aligned} \tfrac{w(v_{a<c})+w(v_{a>c})}{2} &= w_{abcd/}(v) + \tfrac{3}{4}w_{acd/b}(v) + \tfrac{1}{2}w_{cd/ab}(v) + \tfrac{3}{4}w_{abc/d}(v) + \tfrac{1}{2}w_{ac/bd}(v) \\ &\quad + \tfrac{1}{2}w_{c/abd}(v) + \tfrac{1}{2}w_{ab/cd}(v) + \tfrac{1}{2}w_{a/bcd}(v) + \tfrac{1}{2}w_{/abcd}(v) \\ &= w_{abcd/}(v) + \tfrac{3}{2}w_{abc/d}(v) + 3w_{ab/cd}(v) + w_{a/bcd}(v) + \tfrac{1}{2}w_{/abcd}(v). \end{aligned}$$

Minimizing this expression, subject to the equalities of Lemma 3.7, the inequalities of Lemma 3.8, and the fact that the weights of the nine cases sum up to $w(v)$, amounts to solving a linear program. By solving this linear program we get that

$$\frac{w(v_{a<c}) + w(v_{a>c})}{2w(v)} \geq \frac{1}{2}(1 + f_2(\epsilon)) \cdot w(v) \ ,$$

where

$$f_2(\epsilon) = \frac{(3-\epsilon)(1-\epsilon)^3}{(2-\epsilon)^4} \ .$$

It seems intuitively clear that the comparison $a : d$ is a bad comparison from the algorithm's point of view. The adversary will most likely answer with $a > d$. Indeed, by solving the corresponding linear program, we get that

$$\begin{aligned} w(v_{a>d}) &= w_{abcd/}(v) + \tfrac{1}{2}w_{acd/b}(v) + w_{abc/d}(v) + w_{ac/bd}(v) \\ &\quad + \tfrac{1}{2}w_{c/abd}(v) + w_{ab/cd}(v) + w_{a/bcd}(v) + w_{/abcd}(v) \\ &= w_{abcd/}(v) + \tfrac{3}{2}w_{abc/d}(v) + 5w_{ab/cd}(v) + \tfrac{3}{2}w_{a/bcd}(v) + w_{/abcd}(v) \geq \tfrac{3}{4} \ . \end{aligned}$$

As $\frac{1}{2}(1 + f_2(\epsilon)) \leq \frac{3}{4}$, for every $0 \leq \epsilon \leq 1$, we may disregard the comparison $a : d$ from any further consideration.

It is easy to verify that $(1 + f_1(\epsilon))^2 \geq 1 + f_2(\epsilon)$. As a result, we get a lower bound of

$$2n + \frac{n}{8} \cdot \int_0^1 \log_2(1 + f_2(t))dt + o(n) \ \approx \ 2.01227n + o(n) \ .$$

This completes the proof of Theorem 3.6. $\square$

4. Concluding remarks. We presented a reformulation of the $2n + o(n)$ lower bound of Bent and John for the number of comparisons needed for selecting the median of n elements. Using this new formulation we obtained an improved lower bound for pair-forming median finding algorithms. As mentioned, Dor and Zwick [6] have recently extended the ideas described here and obtained a $(2+\epsilon)n$ lower bound for general median finding algorithms, for some tiny $\epsilon > 0$.

We believe that the lower bound for pair-forming algorithms obtained here can be substantially improved. Such an improvement seems to require, however, some new ideas. Obtaining an improved lower bound for pair-forming algorithms may be an important step towards obtaining a lower bound for general algorithms which is significantly better than the lower bound of Bent and John [2].

Paterson [11] conjectures that the number of comparisons required for selecting the median is about $(\log_{4/3} 2)\cdot n \approx 2.41n$.

REFERENCES

[1] Martin Aigner. Producing posets. *Discrete Mathematics*, 35:1–15, 1981.

[2] Samuel W. Bent and John W. John. Finding the median requires $2n$ comparisons. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 213–216, 1985.

[3] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.

[4] Jingsen Chen. *Partial Order Productions.* PhD thesis, Lund University, Box 118, S-221 00 Lund, Sweden, 1993.

[5] Dorit Dor and Uri Zwick. Selecting the median. In *Proceedings of 6th SODA*, pages 88–97, 1995. Journal version in *SIAM Journal on Computing*, 28:1722–1758, 1999.

[6] Dorit Dor and Uri Zwick. Median selection requires $(2+\epsilon)n$ comparisons. In *Proceedings of 37th FOCS*, pages 125–134, 1996. Journal version to appear in *SIAM Journal on Discrete Mathematics.*

[7] Lester R. Ford and Selmer M. Johnson. A tournament problem. *American Mathematical Monthly*, 66:387–389, 1959.

[8] Frank Fussenegger and Harold N. Gabow. A counting approach to lower bounds for selection problems. *Journal of the Association for Computing Machinery*, 26(2):227–238, April 1979.

[9] Donald Ervin Knuth. *The Art of Computer Programming, vol. 3, Searching and Sorting.* Addison-Wesley Publishing Company, Inc., 1973.

[10] I. Munro and P.V. Poblete. A lower bound for determining the median. Technical Report Research Report CS-82-21, University of Waterloo, 1982.

[11] Michael S. Paterson. Progress in selection. In *5th Scandinavian Workshop on Algorithm Theory, Reykjavík, Iceland*, pages 368–379, 1996.

[12] Vaughan R. Pratt and Foong Frances Yao. On lower bounds for computing the i-th largest element. In *14th Annual Symposium on Switching and Automata Theory*, pages 70–81, 1973.

[13] A. Schönhage, M. Paterson, and N. Pippenger. Finding the median. *Journal of Computer and System Sciences*, 13:184–199, 1976.

[14] C.K. Yap. New lower bounds for medians and related problems. Computer Science Report 79, Yale University, 1976. Abstract in *Symposium on Algorithms and Complexity: New Results and Directions*, (J. F. Traub, ed.) Carnegie-Mellon University, 1976.