# LINEAR TIME BOUNDS FOR MEDIAN COMPUTATIONS

by Manuel Blum, Robert W. Floyd, Vaughan Pratt,
Ronald L. Rivest, and Robert E. Tarjan

August 1971

Abstract

New upper and lower bounds are presented for the maximum number of comparisons, $f(i,n)$, required to select the i-th largest of $n$ numbers. An upper bound is found, by an analysis of a new selection algorithm, to be a linear function of $n$:

$$f(i,n) \le 103n/18 < 5.73n \ , \quad \text{for} \quad 1 \le i \le n \ .$$

A lower bound is shown deductively to be:

$$f(i,n) \ge n + \min(i,n-i+1) + \lceil \log_2(n) \rceil - 4 \ , \quad \text{for} \quad 2 \le i \le n-1 \ ,$$

or, for the case of computing medians:

$$f(\lceil n/2 \rceil, n) \ge 3n/2 - 3 \quad .$$

## I. Introduction

New upper and lower bounds are presented for the minimax number of comparisons, $f(i,n)$, required to select the i-th largest of a set $S$ of $n$ distinct numbers. The i-th largest number is that element of $S$ which is less than exactly $i-1$ other elements. Each comparison determines which of two numbers is larger.

An upper bound for $f(i,n)$, linear in $n$, is derived from the analysis of a new selection algorithm in Section II:

$$f(i,n) \le 103n/18 < 5.73n \quad , \quad \text{for} \quad 1 \le i \le n \ .$$

This is essentially an asymptotic bound; for small values of $n$ better results are obtainable with other algorithms. The best previous upper bound was established in 1969 by Hadian and Sobel [2]:

$$f(i,n) \le n - i + (i-1)\lceil \log_2(n-i+2) \rceil \ , \quad \text{for} \quad 1 \le i \le n \ .$$

A lower bound is proved analytically in Section III:

$$f(i,n) \ge n + \min(i,n-i+1) + \lceil \log_2(n) \rceil - 4 \quad , \quad \text{for} \quad 2 \le i \le n-1 \ .$$

The values of $f(i,n)$ for $i = 1$ and $i = n$ are trivial:

$$f(1,n) = f(n,n) = n-1 \quad .$$

In 1932, Josef Schreier [5] showed by construction that:

$$f(2,n) = f(n-1,n) \le n + \lceil \log_2(n) \rceil - 2 \ .$$

S. S. Kislitsin [4] proved in 1965 that Schreier's algorithm was in fact optimal, a result which will be used in Section III. No lower bounds for $f(i,n)$ have previously been published for other values of $i$.

There is no evidence to suggest that either of the new bounds presented is indeed optimal. Thus, it remains an open problem to determine $f(i,n)$ more accurately.

## II. An Upper Bound

In this section an upper bound for $f(i,n)$ is established by analysis of a new selection algorithm. The algorithm is presented first in a simplified form, then in a more efficient version.

The new algorithm operates by discarding all elements of $S$ that are "too large" or "too small", until only the i-th largest element remains. Any element which is:

(i) less than $i$ (or more) other elements, or

(ii) greater than $n-i+1$ (or more) other elements

may clearly be discarded. Approximately one quarter of the remaining elements are discarded on each pass of the algorithm, at a cost per pass proportional to the number of elements remaining -- yielding a total cost proportional to $n$. Previous algorithms have discarded elements one at a time, each at a cost proportional to $\log(n)$ -- yielding a total cost proportional to $n \log(n)$ for finding medians.

The new algorithm is similar to Hoare's selection algorithm [3], in that the basic iteration involves choosing an element $x$ of the set $S$ about which to partition $S$, and then discarding either all those elements larger or all those elements smaller than $x$. The new algorithm, however, selects an element $x$ which is guaranteed to be far from extremal.

### II.A. The Simplified Algorithm

The simplified algorithm iterates five steps until the i-th largest element is found. At each iteration approximately one quarter of the remaining elements are discarded. The algorithm is stated in terms of a fixed (odd) integer $k$, $k \ge 5$, which will be chosen later:

Repeat indefinitely:

1. If $n < 3k$, sort $S$, select the i-th largest, and halt.
2. Arrange $S$ into $\lfloor n/k \rfloor$ columns of $k$ elements each, with $b$ elements left over $(0 \le b < k)$, and sort each column.
3. Find the median, $M$, of the column medians by applying the algorithm recursively.
4. Determine $j$, the number of elements in $S$ that are $\ge M$, by comparing $M$ as needed to every other element (including the $b$ "left-overs").

This research was supported by the National Science Foundation under grant number GJ-992.

5. If $j = i-1$ , halt ($M$ is the i-th largest), else
   if $j > i-1$ , set $n \leftarrow j$ , and discard all elements $\leq M$ , else
   if $j < i-1$ , set $n \leftarrow n-j-1$ , $i \leftarrow i-j-1$ , and discard all elements $\geq M$ .

After step 3, $M$ is less than half of the column medians. Since each column median is less than half of the elements in its column, $M$ is less than approximately $n/4$ elements. Symmetrically, $M$ is also greater than approximately another $n/4$ elements. This can be represented graphically:

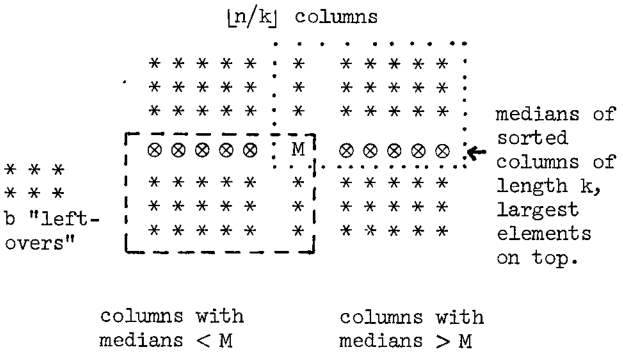


Figure 1

Every element in the dotted box above is $\geq M$ , while every element in the dashed box is $\leq M$ . Therefore at least $n/4$ elements (approximately) are discarded at each iteration.

It is easy to show that no elements are incorrectly discarded in step 5. If $j > i-1$ , then $M$ and every element $< M$ is less than $i$ or more elements. If $j < i-1$ , then $M$ and all elements greater than $M$ are greater than $n-i+1$ or more elements.

Using the following notation the basic algorithm can be analyzed:

$C(n)$ ≡ the maximum number of comparisons required by the basic algorithm to find the i-th largest of $n$ numbers, for all $i$ .

$S(k)$ ≡ the number of comparisons required to sort $k$ numbers.

The algorithm implies:

$$\begin{array}{ll} C(n) = S(n) & \text{for } n < 3k, \text{ else} \\ C(n) \leq \lfloor n/k \rfloor\ S(k) & \text{(for step 2)} \\ \quad + C(\lfloor n/k \rfloor) & \text{(for step 3)} \\ \quad + n-1 & \text{(for step 4)} \\ \quad + C((3n+k)/4) & \text{(for subsequent iterations)} \end{array}$$

For $k \geq 5$ , and $n > k\uparrow 2/(k-4)$ , the sum of the arguments of $Q$ on the right hand side of the latter inequality total less than $n$ , yielding by mathematical induction:

$$C(n) \leq (1+S(k)/k)(4k/(k-4-(k\uparrow 2)/n))n \ ,$$

$$\text{for } k \geq 5,\ n > (k\uparrow 2)/(k-4),$$
$$1 \leq i \leq n \ .$$

Asymptotically, the optimal value of $k$ is 21 , implying:

$$C(n) \leq 348n/17 < 20.5n \qquad (\text{as } n \to \infty) \ .$$

(See [1] for verification that $S(21) = 66$ .) Choosing $k = 21$ , the above inequality is valid for all values of $n$ , since $S(n) < 20n$ for $n < 3k$ .

## II.B. A More Efficient Algorithm

A better upper bound for $f(i,n)$ can be established by modifying the basic algorithm of Section II.A to make it more efficient. The algorithm of this section will be shown to yield the improved bound:

$$f(i,n) \leq 103n/18 < 5.73n \quad .$$

The major changes to the basic algorithms are:

(i) The sorting step (step 2) is done only once, after which the columns are restored to full length by a merge step (step 6) after each pass.

(ii) The partitioning step (step 4) is modified to avoid redundant comparisons and to create a situation where the number of comparisons used is a linear function of the number of elements discarded.

(iii) The discard operation (step 5) is modified to break not more than half the columns, so that the above modifications work well.

The new algorithm will be presented, with steps numbered to correspond to those of the simplified algorithm, and then analyzed. The term "k-columns" will be used to denote sorted columns of length $k$ . The optimal value of $k$ , 15 , will be used throughout for clarity. All sorting operations use the Ford-Johnson algorithm [1].

The new algorithm (selects the i-th largest of a set $S$ of $n$ distinct numbers):

1. If $n < 45$ , sort $S$ , print the i-th largest, and halt.
2. Arrange $S$ into $\lfloor n/15 \rfloor$ 15-columns, with possibly $b$ elements left over. Let $R$ denote the set of "leftovers".
3. Apply the algorithm recursively to find $M$ , the median of the column medians.

At this point, the situation is as follows:

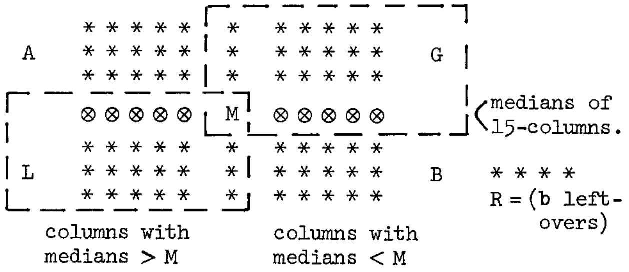


Figure 2

Every element in G is $\geq M$ , every element in L is $\leq M$ , and every element in $A \cup B \cup R$ is of unknown relation to M . Let g(M) and $\ell$(M) denote the number of elements known at any point in time to be greater than M and less than M , respectively.

4(a). Compare each element of R to M .

4(b). If $g(M) > i-1$ or $\ell(M) > n-i$ then go to step 5.

4(c). If $i < n/2$ let X denote A , else B. Partition X about M by inserting M into each 7-column of X using a binary insertion technique (3 comparisons/column).

4(d). If $i < n/2$ let Y denote B , else A . Partition Y about M by inserting M into each column in turn with a linear search technique, beginning with the element nearest each 15-column median and proceeding outwards. Stop as soon as either $g(M) = i$ , $\ell(M) = n-i+1$ , or every 7-column in Y has been partitioned about M .

5. If $g(M) \geq i$ , discard L and all elements in R that are $< M$ . If $\ell(M) \geq n-i$ , discard G and all elements in R that are $> M$ . Decrease n by the number of elements discarded. If $n < 45$ , go to step 1.

In order to restore the conditions to reenter the algorithm at step 3, step 6 merges the remaining columns into 15-columns.

6(a). Let Z denote the set of columns remaining that have length $< 15$ , counting any remaining leftovers as columns of length 1 . Each column in Z is of length $\leq 7$ . The elements in Z will be merged together to form 15-columns. Arrange Z in columns of decreasing length, and separate them into four sections U , P , Q , and R , such that: (Here $|X|$ denotes the number of elements in a set X , x denotes the number of columns.)

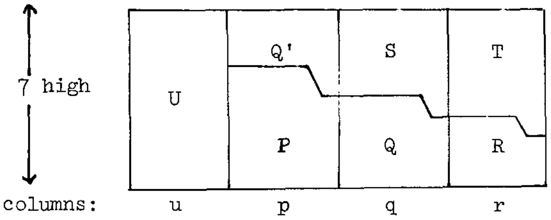
(i) U is the set of all 7-columns.

(ii) $|R| = |Z| / 14$ , and R has the shortest columns.

(iii) P's columns are not shorter than Q's .

(iv) $|P| + |Q| = 7p$ .

Graphically:



Z = U + P + Q + R

Q' + S + T are discarded elements

Figure 3

Here it is easy to see:

$$|Q'| = |Q| = 7p - |P|$$

$$|S| = 7q - |Q| ,$$

$$|T| = 7r - |R| .$$

Now extend each column of P to length 7 by using a binary insertion technique to place each element of Q into a column of P.

6(b). Now every column in Z\R is of length 7 . Merge these 7-columns pairwise to form 14-columns.

6(c). Using binary insertion place each element of R into a 14-column of Z\R . Now Z has been restored to a set of 15-columns. Go to step 3.

This completes the description of the algorithm.

To analyze this algorithm, two lemmas shall be proved and some new notation introduced. Let:

C(n) ≡ an upper bound on the comparisons needed to find the i-th largest of a set of n unordered objects.

C2(n) ≡ an upper bound on the comparisons needed to find the i-th largest of a set of n objects, arranged in 15-columns.

c ≡ the number of comparisons made in step 4(d).

d ≡ the variable component of the number of elements discarded in step 5.

C(n) is the cost of starting at step 1, C2(n) is the cost of starting at step 3.
The Ford-Johnson sorting algorithm [1] requires 42 comparisons to sort 15-elements. Thus,

$$C(n) = 42\lfloor n/15 \rfloor + C2(n) \qquad \text{(step 2)}$$

The following lemma demonstrates the trade-off created between the number of comparisons made in step 4(d) and the number of elements discarded in step 5.

Lemma 1. $c \leq d + n/30$ .

Proof. It is assumed that $i < n/2$ . If $i \geq n/2$ a symmetrical argument applies. There are two cases to consider. In step 5, either elements in G or elements in L are discarded.

Case 1. (G is discarded). In this case d is the number of elements discarded from B . But clearly there can be at most one comparison for every 7-column in B , plus one for every element of B discarded, yielding the lemma.

Case 2. (L is discarded). Let ga be the number of elements in A found to be $> M$ , gb the number of elements in B found to be $> M$ , $\ell a$ the number of elements in A found to be $< M$ , and the number of elements in B found to be $< M$ . Then

$$\begin{aligned} |L| + \ell a + \ell b &\geq n-i \quad \text{(since L is discarded)} \\ &\geq n/2 \quad \text{(since } i < n/2\text{)} \\ &\geq |L| + |B| \\ &\geq |L| + gb + \ell b \quad . \end{aligned}$$

Thus $\ell a \geq gb$ .

But $\ell a = d$ and $gb \leq c + n/30$ (as in case 1), yielding the lemma.

Q.E.D.

The following lemma allows the costs of step 6 to be estimated.

Lemma 2. $|Q| \leq 4d/5$ .

Proof.
$$d = |Q'| + |S| + |T| = |Q| + |S| + |T| \qquad (1)$$
$$|P|/p \geq |Q|/q \geq |R|/r \qquad (2)$$
$$|Q|/p \leq |S|/q \leq |T|/r \qquad (3)$$
$$|Q|/|P| \geq 1/6 \qquad (4)$$
$$14\,|R| = |P| + |Q| + |U| \quad . \qquad (5)$$

Thus
$$\begin{aligned} |Q| &\leq d - |S| - |T| && \text{(from (1))} \\ &\leq d - |S|\ (q+r)/q && \text{(from (3))} \\ &\leq d - |Q|\ (q+r)/p && \text{(from (3))} \\ &\leq d/(1 + q/p + r/p) && \text{(simplifying)} \end{aligned}$$

Also,
$$q/p \geq |Q|/|P| \geq 1/6 \qquad \text{(from (4))}$$

and
$$\begin{aligned} r/p &\geq |R|/|P| && \text{(from (2)} \\ &\geq 7\ |R|/(6(|P| + |Q|)) && \text{(from (4)} \\ &\geq 7\ |R|/(6(|P| + |Q| + |U|)) && \\ &\geq 7/(6(14)) && \text{(from (5))} \\ &\geq 1/12\ . && \end{aligned}$$

Thus,
$$|Q| \leq d/(1 + 1/6 + 1/12) = 4d/5\ .$$

Q.E.D.

The algorithm may now be easily analyzed:

$$\begin{aligned} C2(n) &\leq C(\lfloor n/15 \rfloor) && \text{(step 3(c))} \\ &+ 14 && \text{(step 4(a))} \\ &+ 3(n/30) && \text{(step 4(c))} \\ &+ d + n/30 && \text{(step 4(d))} \\ &+ 12d/5 && \text{(step 6(a))} \\ &+ 13(7n/30 - d)/15 && \text{(step 6(b))} \\ &+ 4(7n/30 - d)/15 && \text{(step 6(c))} \\ &+ C2(\lfloor 11n/15 \rfloor - d) && \text{(iteration)} \end{aligned}$$

Simplifying, and substituting for $C(\lfloor n/15 \rfloor)$ yields:

$$\begin{aligned} C2(n) &\leq 263n(n + (765/263)d)/(90(n + 5d)) \\ &\leq 263n/90\ . \end{aligned}$$

Hence,
$$C2(n) \leq 263n/90 < 2.93n$$

and
$$C(n) \leq 103n/18 < 5.73n\ .$$

## III. A Lower Bound

In this section a lower bound for $f(i,n)$ is derived deductively. The following definitions are useful:

Definition 1. A "partially ordered set" $P = (S,R)$ consists of a set $S$ and a binary relation $R$ on $S$ satisfying the reflexive, antisymmetric, and transitive laws. The statement " $aRb$ " will also be written " $a \leq b$ ".

Definition 2. A "policy" $Q$ is a function from partially ordered sets $P = (S,R)$ to unordered pairs of elements from $S$ , such that $Q(P) = \{a,b\}$ implies $\neg(a \leq b \vee b \leq a)$ . If $R$ is a total order then $Q$ may be undefined.

Clearly a selection procedure can be defined by using a policy repeatedly to specify the next two elements to be compared. If the policy is a computable function, the procedure so defined is an algorithm.

Definition 3. The "extension of a partially ordered set $P = (S,R)$ by an ordered pair $(a,b)$ " is defined to be

$$P' = (S, (R \cup \{(a,b)\})^+)$$

where it is assumed that $\neg(b \leq a)$ and where $R^+$ denotes the transitive closure of $R$ . $P'$ will also be written as $P + (a,b)$ .

Definition 4. A partial ordering $P = (S,R)$ is said to "select the i-th largest of $S$ " if

$$(\exists a \epsilon S)[(i = |\{b|a \leq b\}|) \wedge (n-i+1 = |\{b|b \leq a\}|)].$$

Definition 5. The "cost of a policy $Q$ to select the i-th largest element of $S$ " is defined to be:

$$\hat{Q}(i,P) = 0 \quad \text{if } P \text{ selects the i-th largest of } S, \text{ else}$$
$$1 + \max(\hat{Q}(i, P + (a,b)), \hat{Q}(i, P + (b,a)))$$
$$\text{where } Q(P) = \{a,b\}\ .$$

The definition clearly specifies the intuitive notion of the "worst-case" cost of a selection procedure defined using the given policy. The following theorem shows that there exists an "optimal" policy.

Theorem 1. There exists a policy $Z$ , such that for all finite partial orderings $P = (S,R)$ , and for all $i$ , $1 \leq i \leq |S|$ , and all policies $Q$ :

$$\hat{Z}(i,P) \leq \hat{Q}(i,P)\ .$$

Proof. Clearly $\hat{Q}(i,P)$ is well defined by Definition 5 above, since each comparison adds new information, and the partial ordering must select the i-th largest after at most $n(n-1)/2$ comparisons. Given $i$ and $P$ , a policy $Q$ can only have at most a finite number of values. The policy $Z$ can thus be effectively recursive defined:

$$Z(i,P) \in \{\{a,b\} | \max(\hat{Z}(i, P + (a,b)), \hat{Z}(i, P + (b,a))) \text{ is minimum})\}\ .$$

If more than one value yields minimal cost, any may be chosen. Clearly $f(i,n) = \hat{Z}(i,nil)$ , where $|S| = n$ and nil is the totally unordered set.

Theorem 2. If $P = (S,R)$ and $P' = (S,R')$ are two partially orderings such that $R \subseteq R'$ , then:

$$\hat{Z}(i,P') \leq \hat{Z}(i,P) \quad .$$

Proof. Clearly, Z can do at least as well on P' as on P , since Z could merely specify the same sequence of comparisons as it does on P , skipping any whose result is implied by R' .

Corollary 1. Given a partial ordering $P = (S,R)$ and an $x \in S$ , then

(i) if $\neg(\exists y)(x < y)$ and $i \neq 1$ , then

$$\hat{Z}(i,P) \geq \hat{Z}(i-1,P+\{(y,x)|\forall y \in S\})$$

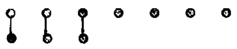
(ii) if $\neg(\exists y)(y < x)$ and $i \neq |S|$ , then

$$\hat{Z}(i,P) \geq \hat{Z}(i,P+\{(x,y)|\forall y \in S\}) \quad .$$

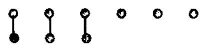
In other words, the cost of the optimal policy is bounded below by its cost when applied to the same partial ordering with the additional information that some element is maximal or minimal. This extremal element can then be effectively discarded, since the fact that it is in known relation to every other element of S implies that it will never be compared against. Discarding a maximal element implies decreasing the value of i by one, however.

A lower bound can now be proved from the above corollary and a detailed consideration of a particular subclass of partial orderings.

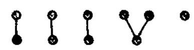
Definition 6. Given a set S , let P(a) stand for that partial ordering of S in which there are 2a elements in pairs, and the remaining elements are uncompared. For example, given $|S| = 10$ , then

$P(3) =$ 

The notation P'(a) shall be used to denote P(a) on a set S' one element smaller than S , e.g.

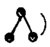
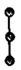
$P'(3) =$ 

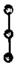
Furthermore, free use of graphics shall be used to indicate extensions of P(a) by adding relations to the uncompared elements, e.g.

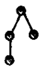
$P(3) +$ $=$ 

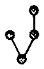
Definition 5 now implies:

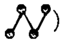
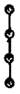
$\hat{Z}(i,P(a)) = 1+\min(\hat{Z}(i,P(a+1))$ ,

$\max(\hat{Z}(i,P(a-1) +$  $),\hat{Z}(i,P(a-1) +$  $)),$

$\max(\hat{Z}(i,P(a-1) +$ $),\hat{Z}(i,P(a-1) +$  $)),$

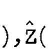
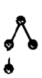
$\hat{Z}(i,P(a-2) +$  $)$ ,

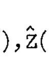
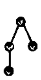
$\hat{Z}(i,P(a-2) +$  $)$ ,

$\max(\hat{Z}(i,P(a-2) +$  $),\hat{Z}(i,P(a-2) +$  $)))$

By application of Theorem 2, each of the max expressions above can be simplified, since the partial ordering in the first argument of each max can be embedded in that of the latter. Thus:

$\hat{Z}(i,P(a)) = 1+\min(\hat{Z}(i,P(a+1)),\hat{Z}(i,P(a-1) +$ $)$ ,

$\hat{Z}(i,P(a-1) +$  $),\hat{Z}(i,P(a-2) +$  $)$ ,

$\hat{Z}(i,P(a-1) +$  $),\hat{Z}(i,P(a-2) +$  $)$ ,

Now, by application of Corollary 1, each of the last five arguments of min above can be bounded below by the cost of the optimal algorithm on a similar partial ordering, one in which new information has been added specifying that one element is maximal (or minimal). The element specified will be one that is already known to be greater than (less than) two other elements. This element may be effectively discarded, since it will never be compared against. Thus

$$\hat{Z}(i,P(a)) \geq$$

$$1+\min(\hat{Z}(i,P(a-1)),\hat{Z}(i-1,P'(a-1)),\hat{Z}(i,P'(a-1))) \quad .$$

Given the boundary condition:

$$\hat{Z}(1,P(0)) = \hat{Z}(n,P(0)) = n-1 \quad ,$$

the following solution can be obtained:

$$\hat{Z}(i,P(a)) \geq n+\min(i,n-i+1) - a - 2 \; , \qquad \text{for } 1 \leq i \leq n \quad .$$

Using Schreier's and Kislitsin's results [4, 5], that

$$\hat{Z}(2,P(0)) = \hat{Z}(n-1,P(0)) = n+\lceil \log_2(n) \rceil - 2 \; ,$$

a slightly improved solution can be derived (for $n > 4$ , $2 \leq i \leq n-1$) :

$$\hat{Z}(i,P(a)) \geq n+\min(i,n-i+1) + \lceil \log_2(n) \rceil - a - 4 \quad .$$

This implies, for finding medians, that

$$f(\lfloor n/2\rfloor, n) = \hat{Z}(\lfloor n/2\rfloor, \mathbf{P}(0)) \geq 3n/2 - 3$$

## References

[1] Ford, Lester R. and Selmer M. Johnson. "A Tournament Problem." American Math. Monthly, Vol. 66, No. 5 (May, 1959).

[2] Hadian, Abdollah, and Milton Sobel. "Selecting the t-th largest using binary errorless comparisons." Technical Report 121, Dept. of Statistics, University of Minnesota. (May, 1969), 15 pp.

[3] Hoare, C. A. R. "Algorithm 65, Find." CACM (July, 1961), 321-322.

[4] Kislitsin, S. S. "On the selection of the k-th element of an ordered set by pairwise comparisons." Sibirsk. Math 5 (1964), 557-564. (MR. 29, No. 2198).

[5] Schreier, Josef. Mathesis Polska 7 (1932), pp. 154-160.