Combinatorial Aspects of C.A.R. Hoare's FIND Algorithm

D.H. Anderson[1]

Operations Research Section,
Civil Aviation Authority, Australia,
GPO Box 367,
Canberra, ACT, 2601

and

R. Brown

CRAY Research (Australia) Pty. Ltd.
283 Normanby Rd.
Port Melbourne, Vic, 3207

Abstract

In his invited address to the 1971 IFIP Congress, Donald E. Knuth [1] examined the average performance of C.A.R. Hoare's FIND algorithm [2] for finding the j-th largest out of n elements. Knuth commented that the recurrence relation that he derived was "not the kind of recurrence that we would normally expect to solve". Nevertheless, he went on to solve it, commenting on the "extraordinary coincidence" of a certain common factor in each of the terms of a derived recurence relation. An extension due to R.C. Singleton [3] to the basic algorithm, known as "median-of-three selection", has seen widespread use in many important computational techniques applied to a diverse set of problems. Because of the renewed interest in median-of-three selection, this paper addresses itself to a serious study of its average performance.

In this paper the median-of-three selection problem is formulated in combinatorial terms and solved by generating function techniques. Closed form expressions are presented for the average of the underlying distribution.

[1]This research was undertaken while this author was employed in the Computer Science Department, University College, University of New South Wales, Australian Defence Force Academy, Canberra, ACT, 2600.

Australasian Journal of Combinatorics 5(1992), pp.109-119

Introduction

C.A.R Hoare's FIND algorithm [2] solves the selection problem, namely that of finding the j-th largest out of a set of n numbers. This problem includes the problem of finding the median of a set of numbers because j can, for example, be chosen to be $\lfloor (n+1)/2 \rfloor$, where $\lfloor\ \rfloor$ is the floor function. Obviously, one way of solving the general problem would be to sort the n numbers and then choose the j-th, but this is clearly doing more work than is necessary. The numbers are not required to be in precise order; all that is required is that the j-th largest be identified. Simple algorithms exist for specific j. For example, it is easy to construct efficient algorithms for $j = 1$ and $j = n$. The FIND algorithm works for general j and is based on the idea of picking a random element, the *pivot*, which is compared with all the remaining elements so that the original set is partitioned into two subsets, one containing numbers less than (or equal to) the pivot, and the other containing values greater than the pivot. Clearly, given the number of elements in each of these two subsets, the problem can be reduced to one involving fewer elements. In his invited address to the 1971 IFIP Congress, Donald E. Knuth [1] examined the average performance of this algorithm. He derived a recurrence relation that he commented was "not the kind of recurrence that we would normally expect to solve", but which he went on to solve because of the fortuitous cancellation of a common factor.

An extension due to Singleton [3] to the basic algorithm is known as median-of-three selection. The pivot element is chosen as the median of the first, middle and last elements of the set. This variant has seen widespread use in recent years. Not only is it the basis of the usual quicksort algorithm but it has also been used in the most efficient of the published methods for least absolute value regression (see, for example, Bloomfield and Steiger [4], Anderson and Steiger [5]).

The renewed interest in median-of-three selection has motivated a serious study of its average performance. As background to the analysis of the median-of-three algorithm and to provide some insights, the basic algorithm will firstly be considered. It will be assumed that all possible orderings are equally likely and, to keep the analysis simpler, the numbers will be assumed to be distinct. With these assumptions, then, the average number of comparisons to find the j-th out of n numbers in the FIND algorithm will be the total over all $n!$ permutations of the number of comparisons to find the j-th value, divided by $n!$.

Consider firstly the case $j = 1$, that is, finding the smallest of the set. When $n = 1$, the pivot is the element required, and this situation can be represented by the graph Γ_1^1 in Figure 1. When $n = 2$ there are two possibilities: either the smaller is chosen as the pivot, in which case the algorithm terminates; or the larger is chosen as pivot, in which case the required number is the smallest out of the remaining one element. This situation is represented graphically by Γ_2^1 of Figure 1. When $n = 3$ there are $3! = 6$ cases, and the graph is Γ_3^1 of Figure 1, where each of the major three branches correspond to two cases each. In general, for $j = 1$ the graph Γ_n^1 of Figure 1 is obtained, with each major branch corresponding to $(n-1)!$ cases.

Similarly, for $j = 2$ the graphs Γ_2^2, Γ_3^2 and Γ_4^2 of Figure 2 can be associated with $n = 2$, $n = 3$ and $n = 4$, respectively, and for $j = 3$ the graphs Γ_3^3 and Γ_4^3 of Figure 3

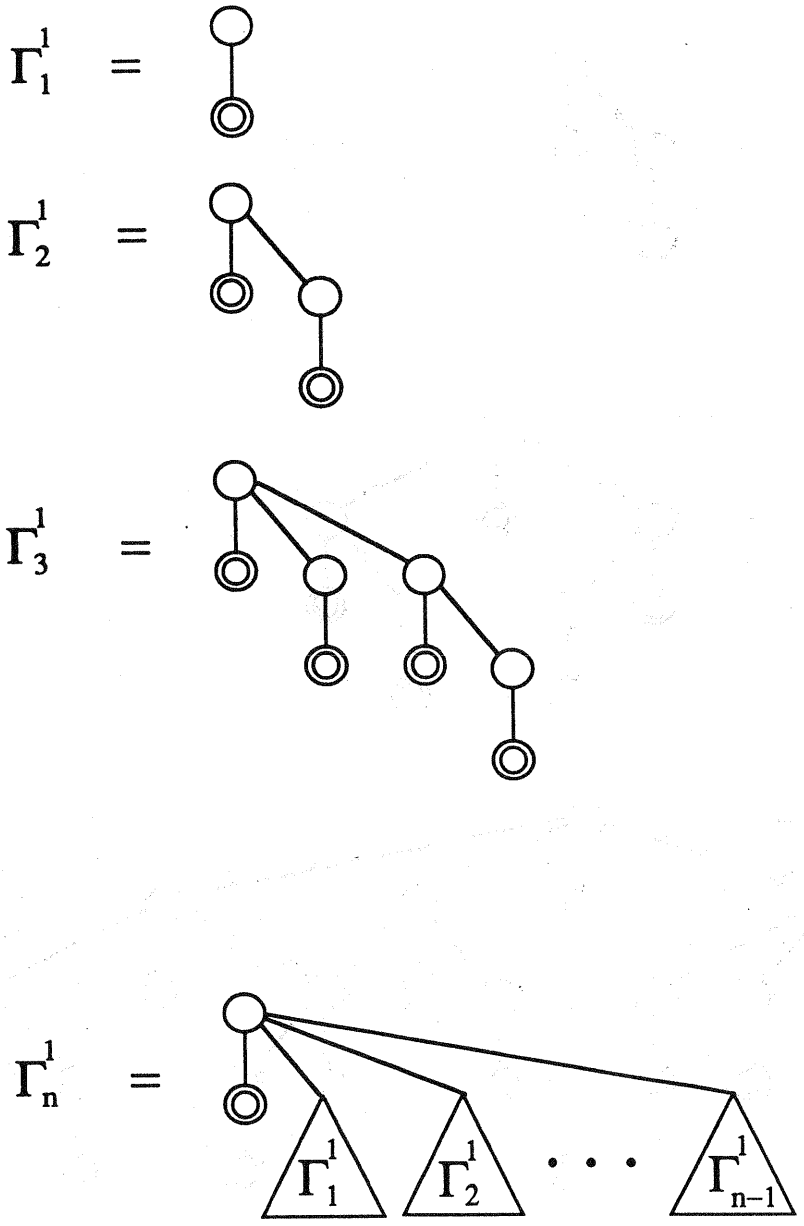

Figure 1: Graphs for selecting the smallest out of n

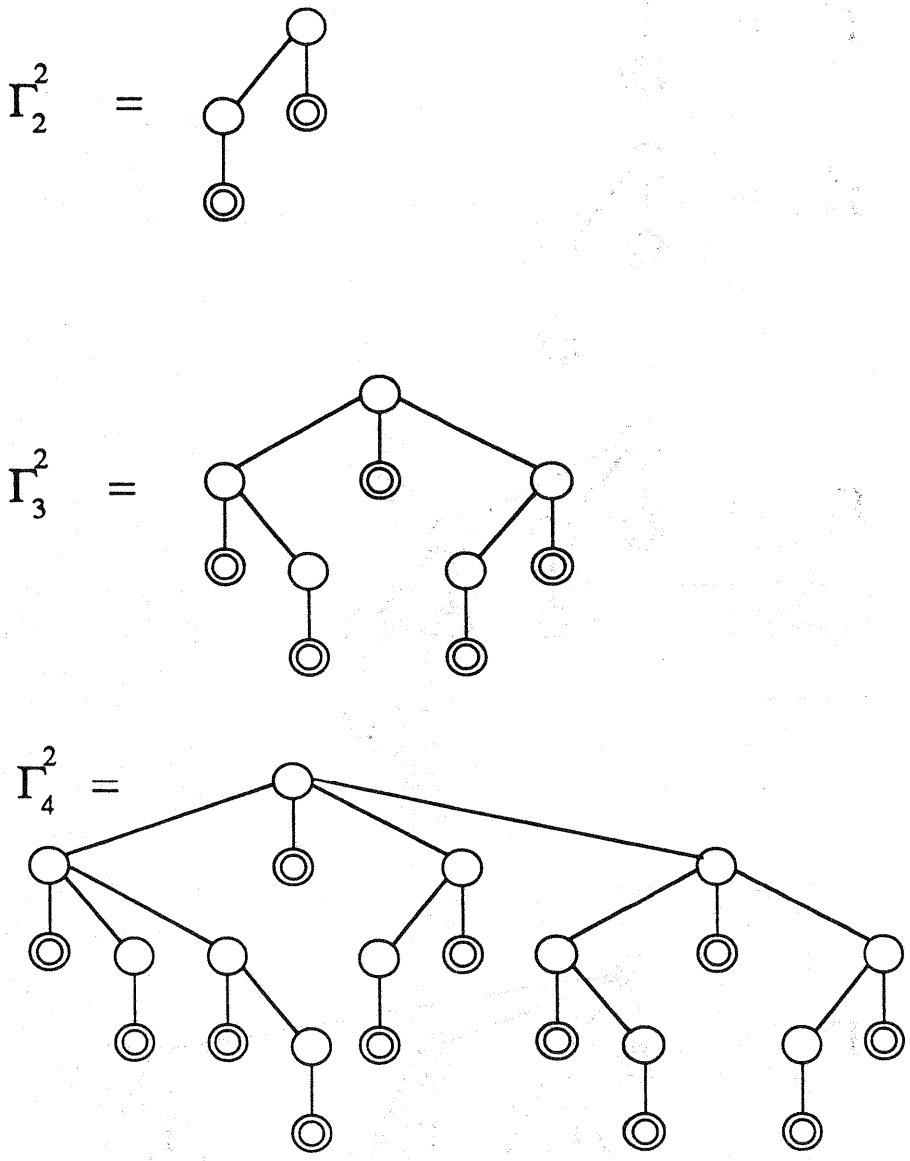

Figure 2: Graphs for selecting the second smallest out of n

can be associated with $n = 3$ and $n = 4$. In general it can be seen that these graphs are of the general form of Γ_n^j of Figure 4. Note that the graphs satisfy the symmetry property that Γ_n^j is the reflection of Γ_n^{n+1-j} in the "GIFT" line that joins the top node to the first terminal node. (If the authors can be forgiven some levity, GIFT is an acronym for Got It First Try.)

Recurrence relations can now be derived for both the basic algorithm and the extension. Although strictly not necessary for the analysis of the basic algorithm, a consideration will be made of the number of "passes". This is actually the simplest of all of the analyses that need to be done, and the difficulty in producing general results in this case demonstrates why the other cases are so intractable.

Number of passes for basic algorithm

In this case, let $G_n^j(x)$ be the polynomial where the coefficient of x^p is the number of times out of the total of $n!$ that the algorithm takes exactly p passes. Assuming, as above, that each permutation is equally likely, and noting that the subgraphs Γ_{n-k}^{j-i} represent $(n-k)!$ arrangements, whereas there are $(n-1)!$ cases for each major branch of Γ_n^j, the relation

$$G_n^j(x) = x\left\{\sum_{k=1}^{j-1}\frac{(n-1)!}{(n-k)!}G_{n-k}^{j-k}(x) + (n-1)! + \sum_{k=j+1}^{n}\frac{(n-1)!}{(k-1)!}G_{k-1}^{j}\right\} \tag{1}$$

is obtained. The second sum can be eliminated to give

$$G_{n+1}^j - (n+x)G_n^j = xn!\sum_{k=1}^{j-1}\left(\frac{G_{n+1-k}^{j-k}}{(n+1-k)!} - \frac{G_{n-k}^{j-k}}{(n-k)!}\right) \tag{2}$$

where $G_1^1 = x$ and the explicit dependence of G on x has been dropped.

Unfortunately, even in this very simple case, a closed-form solution for $G(x)$ is not known, although a great deal of progress can be made. If attention is restricted to the total over all the permutations of the number of passes, denoted by λ_n^j, then

$$\lambda_n^j = \frac{d}{dx}G_n^j(x)\bigg|_{x=1}. \tag{3}$$

Differentiating the relation 2 with respect to x, putting $x = 1$, and noting that $G_n^j(1) = n!$, gives the relation

$$\lambda_{n+1}^j - (n+1)\lambda_n^j = n!\left\{1 + \sum_{k=1}^{j-1}\left(\frac{\lambda_{n+1-k}^{j-k}}{(n+1-k)!} - \frac{\lambda_{n-k}^{j-k}}{(n-k)!}\right)\right\} \tag{4}$$

which can be solved reasonably easily to give

$$\lambda_n^j = n!\,(H_{n+1-j} + H_j - 1) \tag{5}$$

where H_n is the harmonic sum $\sum_{k=1}^{n} 1/k$.

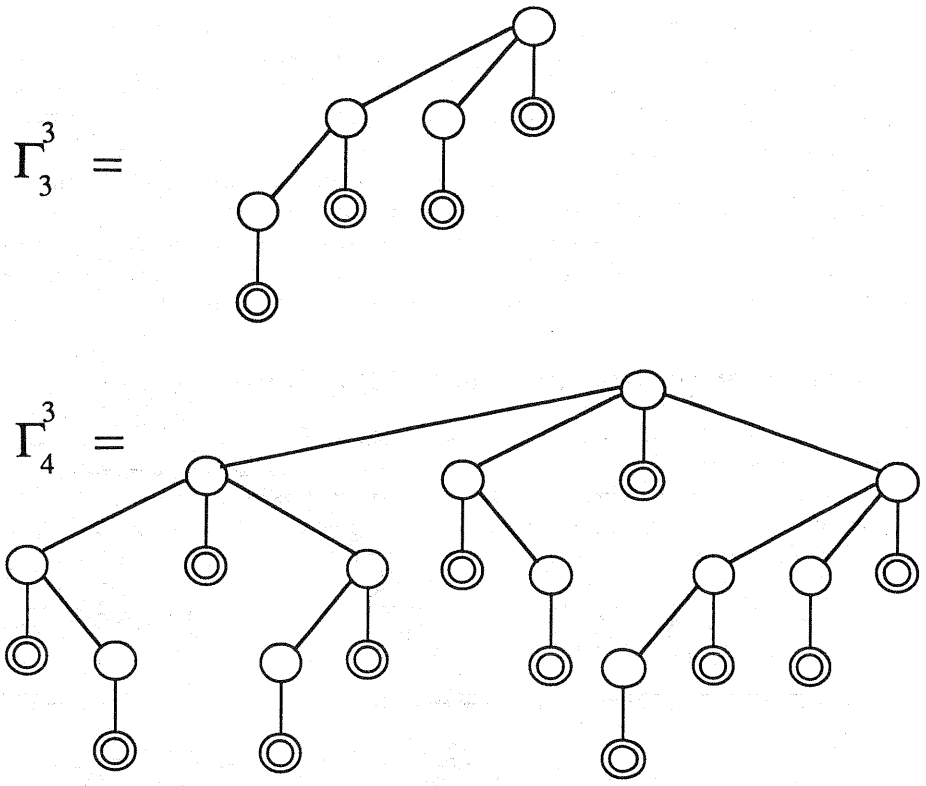

Figure 3: Graphs for selecting the third smallest out of n

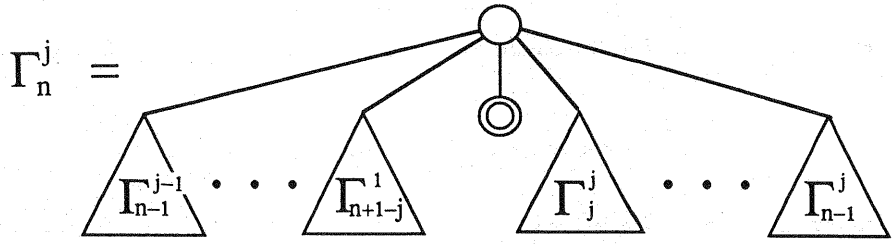

Figure 4: The general case, selecting the j-th out of n

Number of comparisons for basic algorithm

This is the next most difficult problem. The recurrence relation for the generating function undergoes a minor change:

$$G_n^j(x) = x^{n-1}\left\{\sum_{k=1}^{j-1}\frac{(n-1)!}{(n-k)!}G_{n-k}^{j-k} + (n-1)! + \sum_{k=j+1}^{n}\frac{(n-1)!}{(k-1)!}G_{k-1}^{j}\right\} \tag{6}$$

which becomes

$$G_{n+1}^j - x(n+x^{n-1})G_n^j = x^n n!\sum_{k=1}^{j-1}\left(\frac{G_{n+1-k}^{j-k}}{(n+1-k)!} - \frac{G_{n-k}^{j-k}}{(n-k)!}\right) \tag{7}$$

and now obtaining a general solution seems an impossible goal. Instead, if the last relation is differentiated at $x = 1$ and μ_n^j denotes $\frac{d}{dx}G_n^j(x)\Big|_{x=1}$, a relation analogous to 4 is obtained for the total number of comparisons over all permutations:

$$\mu_{n+1}^j - (n+1)\mu_n^j = n!\left\{2n + \sum_{k=1}^{j-1}\left(\frac{\mu_{n+1-k}^{j-k}}{(n+1-k)!} - \frac{\mu_{n-k}^{j-k}}{(n-k)!}\right)\right\} \tag{8}$$

The solution is a litle more difficult to obtain than previously. One approach is to define

$$T_n^j = \frac{\mu_{n+1}^j}{(n+1)!} - \frac{\mu_n^j}{n!}, \tag{9}$$

solve the resulting equation, and then obtain $\mu_n^j/n!$. The result is

$$\mu_n^j = 2n!\left\{n+3+(n+1)H_n - (j+2)H_j - (n+3-j)H_{n+1-j}\right\}, \tag{10}$$

which agrees with the result in Knuth's IFIP paper [1].

Consider, now, the median-of-three extension. The analysis is complicated by the fact that even though all permutations are considered equally likely, the number of cases corresponding to each major branch of Γ_n^j are not equal. If the branches are numbered from left to right according to $k = 1, 2, \cdots, n$ then the number of cases corresponding to the k-th branch is

$$6(k-1)(n-k)(n-3)!$$

The k-th branch corresponds to the situation where the k-th largest number is chosen as the pivot. Since the pivot is the median of three elements, there are $k-1$ possibilities for the smallest of the three, and $n-k$ possibilities for the largest. The factor 6 takes into account the arrangement of the three elements. If the pivot is the k-th largest number, then, after the partitioning process, if $k < j$ the remaining problem will be to find the $(j-k)$-th largest out of $n-k$ elements, and when $k > j$ it will be to find the j-th largest out of $k-1$ elements.

Number of passes in Median-of-three algorithm

The generating function in this case is

$$\begin{aligned} G_n^j &= x\left\{\sum_{k=1}^{j-1} G_{n-k}^{j-k}\frac{6(k-1)(n-k)(n-3)!}{(n-k)!}\right. \\ &\quad +6(j-1)(n-j)(n-3)! \\ &\quad \left.+\sum_{k=j+1}^{n} G_{k-1}^{j}\frac{6(k-1)(n-k)(n-3)!}{(k-1)!}\right\} \end{aligned} \tag{11}$$

which can be rewritten in the form

$$\begin{aligned} &G_{n+2}^j - 2(n-1)G_{n+1}^j + ((n-1)(n-2)-6x)\,G_n^j = \\ &\quad 6x(n-1)!\sum_{k=1}^{j-1}\frac{k-1}{(n+1-k)!}\left\{G_{n+2-k}^{j-k} - 2(n+1-k)G_{n+1-k}^{j-k}+\right. \\ &\qquad \left.(n+1-k)(n-k)G_{n-k}^{j-k}\right\}. \end{aligned} \tag{12}$$

The more complicated nature of this relation makes it quite intractable. As before, however, it can be differentiated at $x = 1$. If λ_n^j denotes $\frac{d}{dx}G_n^j(x)\Big|_{x=1}$, then the resulting relation is

$$\begin{aligned} &\lambda_{n+2}^j - 2(n-1)\lambda_{n+1}^j + (n+1)(n-4)\lambda_n^j = \\ &\quad 6n! + 6(n-1)!\sum_{k=2}^{j-1}\frac{k-1}{(n+1-k)!}\left\{\lambda_{n+2-k}^{j-k} - 2(n+1-k)\lambda_{n+1-k}^{j-k}+\right. \\ &\qquad \left.(n+1-k)(n-k)\lambda_{n-k}^{j-k}\right\}. \end{aligned} \tag{13}$$

Although this relation looks quite difficult to deal with, close examination indicates that the solution is of the form

$$\lambda_n^j = \frac{6}{5}n!H_n + \sum_{k=0}^{j}\theta_k^j\cdot(n-k)! + \sum_{k=j-4}^{j-1}\phi_k^j\cdot[-4]^{n-k} \tag{14}$$

where $[a]^n$ indicates the rising factorial $a(a+1)\cdots(a+n-1)$. Note that the last term disappears for large n (as long as j is not too big), so it will be ignored.

Consider the θ_k^j in the first summation. Solutions for some special cases yield the following results, where the notation $[a]_n$ indicates the falling factorial $a(a-1)\cdots(a+1-n)$,

$$\begin{aligned} \theta_1^j &= 0 \\ \theta_2^j &= -\frac{3}{5}[j-1]_2 \\ \theta_3^j &= -\frac{2}{5}[j-1]_3 \\ \theta_4^j &= -\frac{3}{10}[j-1]_4 \end{aligned}$$

$$\theta_5^j = \frac{6}{25}[j]_5$$
$$\theta_6^j = -\frac{1}{5}[j-1]_6.$$

Although it seems possible that a general pattern may exist, it transpires that the first six θ_k^j values are special cases, and, in general

$$\theta_k^j = \frac{6}{7k}\left(\frac{2}{5}[j]_k - [j-1]_k\right) \text{ for } k \geq 7 \tag{15}$$

To complete the analysis of the median-of-three algorithm, the number of comparisons *after* the pivot selection is required. The choice of the pivot itself requires 3 comparisons in most cases and 2 in others. The average is 8/3 comparisons per pivot selection.

Number of comparisons in Median-of-three algorithm after pivot selection

The generating function here is the same as for the number of passes except that the factor x is replaced by x^{n-3}. Differentiating and putting $x = 1$ and $\mu_n^j = \frac{d}{dx}G_n^j\Big|_{x=1}$ gives the relation

$$\mu_{n+2}^j - 2(n-1)\mu_{n+1}^j + (n+1)(n-4)\mu_n^j =$$
$$12(n-1)n! + 6(n-1)!\sum_{k=2}^{j-1}\frac{k-1}{(n+1-k)!}\left\{\mu_{n+2-k}^{j-k} - 2(n+1-k)\mu_{n+1-k}^{j-k} + (n+1-k)(n-k)\mu_{n-k}^{j-k}\right\}. \tag{16}$$

This relation looks even more difficult to deal with than the previous one, however, close examination indicates that the solution is of the form

$$\mu_n^j = 2(n+1)! - \frac{24}{5}n!H_n + \sum_{k=0}^{j}\alpha_k^j\cdot(n-k)! + \sum_{k=j-4}^{j-1}\beta_k^j\cdot[-4]^{n-k}. \tag{17}$$

The last term again disappears for large enough n and will again be ignored. Consideration of the α_k^j in the first summation yields the following special cases

$$\alpha_1^j = -3[j-1]_2$$
$$\alpha_2^j = \frac{12}{5}[j-1]_2$$
$$\alpha_3^j = \frac{8}{5}[j-1]_3$$
$$\alpha_4^j = \frac{6}{5}[j-1]_4$$
$$\alpha_5^j = \frac{6}{5}[j-1]_4 + \frac{36}{25}[j-1]_5$$
$$\alpha_6^j = \frac{4}{5}[j-1]_6$$

and the general solution

$$\alpha_k^j = \frac{12}{35}\left([j-1]_{k-1} + \frac{16}{k}[j-1]_k\right) \text{ for } k \geq 7 \tag{18}$$

Conclusion

The total number of comparisons is

$$\frac{8}{3}\lambda_n^j + \mu_n^j$$

Note some minor correction is required due to the fact that sometimes there are only 1 or 2 elements, and the median-of-three selection algorithm then requires special processing. This correction is less than 1 comparison per permutation.

The exact expression for the total number of comparisons is complicated by the special forms for θ_k^j and α_k^j when $k \leq 6$, however, a very accurate approximation is given by

$$2n!\left(n + \frac{44}{35}H_n - \frac{72}{35}H_{n+1-j} - \frac{72}{35}H_j + \frac{3}{2}\frac{j(n+1-j)}{n}\right). \tag{19}$$

A comparison between this expression and the one for the basic algorithm is given in Table 1 for $n = 1000$. The Table shows that the median-of-three modification can be expected to be better than the basic algorithm in all cases except when finding the smallest (or largest) of the set. When the median is required or an order statistic not too close to the first or last an expected improvement of 15 to 20 percent should be achievable.

j	Simple Selection	Median-of-three Selection	Relative improvement (percent)
1	1985	1987	-0.1
2	1996	1988	0.4
10	2073	2006	3.2
50	2352	2112	10.2
100	2603	2237	14.1
200	2952	2445	17.2
300	3172	2595	18.2
400	3296	2684	18.6
500	3336	2714	18.7

Table 1: Average number of comparisons to find the j-th largest out of a set of 1000 numbers.

References

1. Knuth, D.E., Mathematical Analysis of Algorithms, in *Information Processing 71*, Proceedings of the 1971 IFIP Congress, North-Holland, Amsterdam, 1972, 19–27.

2. Hoare, C.A.R., FIND (Algorithm 65), *Comm. ACM*, 1961, 321–322.

3. Singleton, R.C. SORT (Algorithm 347), *Comm. ACM*, 1969, 185–186.

4. Bloomfield, P and Steiger, W.L., Least absolute deviations curve-fitting, *SIAM J. Scientific and Statistical Comp.*, 1980, 290–301.

5. Anderson, D.H. and Steiger, W.L., A Comparison of Methods for Discrete L_1 Curve-fitting, Department of Computer Science Report DCS-TR-96, Rutgers University, New Brunswick, New Jersey, 1981.