Alcom-FT Technical Report Series
ALCOMFT-TR-01-187

Analysis of Quickfind with small subfiles *

Conrado Martínez† Daniel Panario‡ Alfredo Viola§

Abstract

In this paper we investigate the variants of the well-known Hoare's Quickfind algorithm for the selection of the j-th element out of n, when recursion stops for subfiles whose size is below a predefined threshold and a simpler algorithm is run instead. We provide estimates for the combined number of passes, comparisons and exchanges, for both the basic quickfind and median-of-three quickfind. In each case, we consider two policies for the small subfiles: insertion sort and selection sort, but the analysis could be easily adapted for alternative policies. We obtain the average cost for each of these variants and compare them with the costs of standard variants which do not use cutoff. We also give the best explicit cutoff bound for each of the variants.

1 Introduction

Hoare's quickfind [2] selects the j-th element (equivalently, the element of *rank* j in ascending order, the j-th order statistic) out of an array of n elements by picking an element from the array —the pivot— and rearranging the array so that elements smaller than the pivot are to its left and elements larger than the pivot are to its right. If the pivot has been brought to position $k = j$ then it is the sought element; otherwise, if $j < k$ then the procedure is recursively applied to the subarray to the left of the pivot, and if $j > k$ the process continues in the right subarray. A similar principle is used in the celebrated quicksort algorithm [3], also by Hoare; once the pivot

*The first author was supported by project DGES PB98-0926 (AEDRI) of the Spanish Ministry for Education and Science and the Future and Emergent Technologies programme of the EU under contract IST-1999-14186 (ALCOM-FT). The second author was supported by NSERC of Canada (grant # 238757-01). The third author was supported by the CSIC of Uruguay (project *Desarrollo de métodos matemáticos para analizar algoritmos y su aplicación al análisis de algoritmos criptográficos*).

†Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya. E-08034, Barcelona, Spain. `conrado@lsi.upc.es`.

‡School of Mathematics and Statistics. Carleton University. K1S 5B6, Ottawa, Canada. `daniel@math.carleton.ca`.

§Instituto de Computación. Universidad de la República. Casilla de Correo 16120, Distrito 6, Montevideo, Uruguay. `viola@fing.edu.uy`.

is brought into place by the partitioning of the array, the subarrays to its left and right are recursively sorted.

Quickfind performs well in practice, its average cost being linear. Knuth [5] has shown that the average number of comparisons $C_{n,j}$ needed to locate the j-th element out of n is

$$C_{n,j} = 2((n+1)H_n - (n+3-j)H_{n+1-j} - (j+2)H_j + n + 3),$$

where $H_n = \sum_{1\leq i\leq n} 1/i = \log n + \mathcal{O}(1)$ denotes the n-th harmonic number. Clearly, $C_{n,j}$ is $\Theta(n)$ for any j, $1 \leq j \leq n$.

In order to make uneven partitions leading to $\Theta(n^2)$ worst-case performance more unlikely, one should use the median of a small sample of s elements as the pivot of each recursive stage. The median-of-three variant of quickfind yields a substantial improvement over the standard variant [1, 4] reducing the average number of comparisons and effectively diminishing the probability of worst-case input. For any j, the average cost of quickfind with median-of-three is less than or equal to the average cost of the standard variant. Averaging over all j, the expected number of comparisons made by quickfind is $C_n = 3n + o(n)$, whereas it reduces to $C_n = 5n/2 + o(n)$ when the median-of-three strategy for pivot selection is used.

Besides this major, but rather specific, optimization of the algorithm, other general optimization techniques should be used in a carefully engineered implementation of quickfind. For instance, recursion removal and loop unwrapping; the former is straightforward, because for quickfind we have tail recursion. Another general optimization for divide-and-conquer algorithms is *recursion cutoff*: for sufficiently small inputs, we switch from the recursive divide-and-conquer algorithm to a simpler algorithm. For instance, it is well known that switching from quicksort to insertion sort when subfiles have $n_0 = 9$ elements improves the overall performance of the algorithm. Cutoff values in the range 6 to 15 do about as well, but $n_0 = 9$ is the optimal choice for most implementations. Actually, for quicksort, it is typically better to ignore small subfiles and perform a single pass of insertion sort over the entire file [7, 8].

In this paper, we tackle the analysis of quickfind with recursion cutoff, for both the standard and the median-of-three variants. Computing the optimal cutoff value can be readily done using dynamic programming, solving the recurrences for "small" values of the parameters. However, this approach does not guarantee success if the cost function were not increasing with n or attained several minimal values for different values of the cutoff size n_0. Also, in this paper we want to concentrate in the methodological and analytical aspects of the problem. Our approach, although more complex and difficult to apply, yields much more information about the problem at hand.

We consider using insertion sort and selection sort for small subfiles. Actually, in the case of selection sort we consider a variant which stops as soon as the desired element is found (unlikely insertion sort, which must sort the whole subfile in order to locate the j-th element); also, it starts looking for the j-th element from the most suitable extreme: namely, if $j \leq n/2$ then it scans the array from left to right, whereas if $j > n/2$ then it scans the array from right to left. We perform our analysis trying to keep it as general as possible, in order to facilitate a similar analysis using other algorithms for small subfiles. This analysis can also be generalized for the median-of-three variant.

The original motivation of our analysis is to determine the optimal choices of cutoff values n_0 for each combination of small subfiles algorithm and pivot selection strategy. Since recursion cutoff affects the lower order terms of the performance of the algorithm, not its main linear order term, it only makes sense to consider the expected *total cost of the algorithm*, not just the expected number of comparisons. Therefore, we work with a measure of cost that combines the cost of comparisons, exchanges and passes. As a result of our initial research goals, we shifted the original motivation to something more ambitious, namely, the study of general techniques for the analysis of recursive algorithms with cutoff.

The paper is organized as follows. The methodology used in this paper is based on solving recurrences via generating functions and related differential equations. This framework is summarized in Section 2. The general approach for analyzing quickfind with small subfiles is presented in Section 3. The required functions when we use insertion sort and selection sort for small subfiles are given in Section 4. Tables providing explicit cutoff bounds for the studied variants are in Section 5. Finally, further studies are addressed in Section 6.

2 Basic methodology

Our methodology is based upon the translation of recurrence relations into functional equations over corresponding generating functions. This is typical in this area of research (see for instance [4]). The analysis of quickfind and its variants leads systematically to differential equations. In many instances, we can solve them exactly or we can provide suitable asymptotic estimates of the solution near the dominant singularities. This general framework is shortly described below.

As an example, we start by establishing a recurrence for the probability generating function (pgf) of the number of comparisons in quickfind. Let $\mathbf{C}_{n,j}$ denote the random variable "number of comparisons to select the j-th element out of n made by quickfind". Then we have the following recurrence

$$\begin{aligned}\mathbf{C}_{n,j}(z) &= \sum_{k\geq 0} \Pr\{\mathbf{C}_{n,j} = k\}\, z^k \qquad (1)\\ &= z^c \left(\sum_{k=1}^{j-1} \pi_{n,k} \mathbf{C}_{n-k,j-k}(z) + \pi_{n,j} + \sum_{k=j+1}^{n} \pi_{n,k} \mathbf{C}_{k-1,j}(z) \right),\end{aligned}$$

where $\pi_{n,k}$ is the probability of the k-th element (in sorted order) being selected as the pivot, and c is the number of comparisons made in a single partitioning phase of quickfind. Typically, we use the convention of [4] and take $c = n-1$. Exchanges and passes satisfy similar recurrences; for instance, it suffices to set $c = 1$ above to get the recurrence satisfied by the pgf for the number of passes.

If we could solve the recurrence above, then it would be very easy to derive the average and the variance (and in general the i-th moment) of the number of comparisons. This turns out to be very difficult; however, we can readily obtain a recurrence for the average (or any factorial moment) by differentiating both sides of the recurrence and setting $z = 1$ (see for instance [10]). Applying this procedure to the recurrence (1) we find the following recurrence for $C_{n,j}$, the average number of comparisons, which is equal to $\mathbf{C}'_{n,j}(1)$:

$$C_{n,j} = c + \sum_{k=1}^{j-1} \pi_{n,k} C_{n-k,j-k} + \sum_{k=j+1}^{n} \pi_{n,k} C_{k-1,j}, \qquad (2)$$

where we assume that $C_{n,j} = 0$ whenever $j < 1$ or $j > n$.

The general method for solving the recurrences like (2) is as follows: (i) multiply both sides by some appropriate quantity ("integration factor"); (ii) take the summation over all values of n in some domain in which the recurrence is true; (iii) define a generating function and try to find a functional equation satisfied by this generating function (recurrences like (2) lead to differential equations); and finally, (iv) solve the differential equation and extract coefficients from the generating function.

3 Quickfind with small subfiles

In this section, we give the study of the standard quickfind (a random pivot is chosen from the elements in the current subfile) when we use a different policy for the small subfiles. An interesting feature of our approach is to provide a common presentation for all the costs involved. In the next sections we particularize these results to a certain model of costs and consider two different algorithms, namely insertion and selection sort, for small subfiles.

One can easily verify that for all $1 \leq j \leq n$, the following recurrence relation for the average cost $C_{n,j}$ when selecting the j-th out of n holds:

$$C_{n,j} = \begin{cases} t_{n,j} + \sum_{k=1}^{j-1} \pi_{n,k} C_{n-k,j-k} + \sum_{k=j+1}^{n} \pi_{n,k} C_{k-1,j} & \text{if } n > n_0, \\ b_{n,j} & \text{if } n \leq n_0, \end{cases} \tag{3}$$

where $t_{n,j}$ is the average cost of a single pass of the algorithm (for instance, the combined cost of the selection of the pivot, the partitioning of the subfile and eventually any associated bookkeeping), and $b_{n,j}$ is the same average cost but for the small subfiles algorithm to find the j-th out of n.

Let us assume that we use the basic algorithm for $n > n_0$, so $\pi_{n,k} = \frac{1}{n}$ for all $1 \leq k \leq n$. We take $C_{n,j} = 0$ for all values n, j such that either $j < 1$ or $j > n$. Now, we consider two cases: $n_0 \geq j$ and $n_0 < j$.

If $j \leq n_0$, we multiply both sides of (3) by nz^{n-1} and sum over all $n > n_0$ to obtain

$$\begin{aligned} \sum_{n=n_0+1}^{\infty} nC_{n,j} z^{n-1} = \sum_{n=n_0+1}^{\infty} nt_{n,j} z^{n-1} &+ \sum_{n=n_0+1}^{\infty} \sum_{k=1}^{j-1} C_{n-k,j-k} z^{n-1} \\ &+ \sum_{n=n_0+1}^{\infty} \sum_{k=j+1}^{n} C_{k-1,j} z^{n-1}. \end{aligned}$$

We define the generating function $C_j(z) = \sum_{n \geq 0} C_{n,j} z^n$. Therefore, the equation above implies:

$$\begin{aligned} C'_j(z) &= \sum_{n \geq 1} nC_{n,j} z^{n-1} = \sum_{n=j}^{n_0} nC_{n,j} z^{n-1} + \sum_{n=n_0+1}^{\infty} nC_{n,j} z^{n-1} \\ &= \sum_{n=j}^{n_0} nb_{n,j} z^{n-1} + \sum_{n=n_0+1}^{\infty} nt_{n,j} z^{n-1} \\ &\quad + \sum_{k=1}^{j-1} \sum_{n=n_0+1}^{\infty} C_{n-k,j-k} z^{n-1} + \sum_{k=j+1}^{\infty} \sum_{n=\max(n_0+1,k)}^{\infty} C_{k-1,j} z^{n-1}. \end{aligned}$$

Considering the third sum in the previous equation for $n \geq j$, we have

$$\begin{aligned} C'_j(z) = \sum_{n=j}^{n_0} nb_{n,j} z^{n-1} &+ \sum_{n=n_0+1}^{\infty} nt_{n,j} z^{n-1} + \sum_{k=1}^{j-1} z^{k-1} \sum_{n=j}^{\infty} C_{n-k,j-k} z^{n-k} \\ &- \sum_{k=1}^{j-1} \sum_{n=j}^{n_0} C_{n-k,j-k} z^{n-1} + \sum_{k=j+1}^{\infty} C_{k-1,j} \frac{z^{\max(n_0+1,k)-1}}{1-z}. \end{aligned}$$

Writing the third summand in terms of $C_{j-k}(z)$ and simplifying the fourth term, we obtain

$$\begin{aligned}C_j'(z) &= \sum_{n=j}^{n_0} nb_{n,j}z^{n-1} + \sum_{n=n_0+1}^{\infty} nt_{n,j}z^{n-1} + \sum_{k=1}^{j-1} z^{k-1}C_{j-k}(z) \\ &\quad - \sum_{k=1}^{j-1}\sum_{n=j}^{n_0} b_{n-k,j-k}z^{n-1} + \sum_{k=j+1}^{n_0+1} C_{k-1,j}\frac{z^{n_0}}{1-z} + \sum_{k=n_0+2}^{\infty} C_{k-1,j}\frac{z^{k-1}}{1-z}.\end{aligned}$$

Simplifying the last two terms, we have

$$\begin{aligned}C_j'(z) &= \sum_{n=j}^{n_0} nb_{n,j}z^{n-1} + \sum_{n=n_0+1}^{\infty} nt_{n,j}z^{n-1} + \sum_{k=1}^{j-1} z^{k-1}C_{j-k}(z) \\ &\quad - \sum_{k=1}^{j-1}\sum_{n=j}^{n_0} b_{n-k,j-k}z^{n-1} + \frac{z^{n_0}}{1-z}\sum_{k=j}^{n_0} b_{k,j} + \frac{1}{1-z}\left(C_j(z) - \sum_{k=j}^{n_0} C_{k,j}z^k\right).\end{aligned}$$

Finally, we obtain

$$\begin{aligned}C_j'(z) &= \sum_{n=j}^{n_0} nb_{n,j}z^{n-1} + \sum_{n=n_0+1}^{\infty} nt_{n,j}z^{n-1} + \sum_{k=1}^{j-1} z^{k-1}C_{j-k}(z) \\ &\quad - \sum_{k=1}^{j-1}\sum_{n=j}^{n_0} b_{n-k,j-k}z^{n-1} + \frac{z^{n_0}}{1-z}\sum_{k=j}^{n_0} b_{k,j} + \frac{C_j(z)}{1-z} - \frac{1}{1-z}\sum_{k=j}^{n_0} b_{k,j}z^k.\end{aligned}$$

Therefore, for all $j \le n_0$, we have

$$C_j'(z) = \sum_{k=1}^{j-1} z^{k-1}C_{j-k}(z) + \frac{C_j(z)}{1-z} + f_j^{\mathrm{low}}(z), \tag{4}$$

where

$$\begin{aligned}f_j^{\mathrm{low}}(z) &= \sum_{n=j}^{n_0} nb_{n,j}z^{n-1} + \sum_{n=n_0+1}^{\infty} nt_{n,j}z^{n-1} - \sum_{k=1}^{j-1}\sum_{n=j}^{n_0} b_{n-k,j-k}z^{n-1} \\ &\quad + \frac{z^{n_0}}{1-z}\sum_{k=j}^{n_0} b_{k,j} - \frac{1}{1-z}\sum_{k=j}^{n_0} b_{k,j}z^k.\end{aligned}$$

Similarly, for $j > n_0$, we multiply both sides of (3) by nz^{n-1} and sum for all $n \geq j$:

$$
\begin{aligned}
C_j'(z) &= \sum_{n=j}^{\infty} n t_{n,j} z^{n-1} + \sum_{n=j}^{\infty} \sum_{k=1}^{j-1} C_{n-k,j-k} z^{n-1} + \sum_{n=j}^{\infty} \sum_{k=j+1}^{n} C_{k-1,j} z^{n-1} \\
&= \sum_{n=j}^{\infty} n t_{n,j} z^{n-1} + \sum_{k=1}^{j-1} z^{k-1} \sum_{n=j}^{\infty} C_{n-k,j-k} z^{n-k} + \sum_{k=j+1}^{\infty} \sum_{n=k}^{\infty} C_{k-1,j} z^{n-1} \\
&= \sum_{n=j}^{\infty} n t_{n,j} z^{n-1} + \sum_{k=1}^{j-1} C_{j-k}(z) z^{k-1} + \sum_{k=j+1}^{\infty} C_{k-1,j} \frac{z^{k-1}}{1-z} \\
&= \sum_{n=j}^{\infty} n t_{n,j} z^{n-1} + \sum_{k=1}^{j-1} C_{j-k}(z) z^{k-1} + \frac{C_j(z)}{1-z}.
\end{aligned}
$$

Therefore all $j > n_0$, we have

$$
C_j'(z) = \sum_{k=1}^{j-1} z^{k-1} C_{j-k}(z) + \frac{C_j(z)}{1-z} + f_j^{\mathrm{high}}(z), \tag{5}
$$

where $f_j^{\mathrm{high}}(z) = \sum_{n=j}^{\infty} n t_{n,j} z^{n-1}$. If we define $g_j(z)$ as follows

$$
g_j(z) = \begin{cases} f_j^{\mathrm{low}}(z) & \text{if } j \leq n_0, \\ f_j^{\mathrm{high}}(z) & \text{if } j > n_0, \end{cases} \tag{6}
$$

then, (4) and (5) imply that for all $j \geq 1$,

$$
C_j'(z) = \sum_{k=1}^{j-1} z^{k-1} C_{j-k}(z) + \frac{C_j(z)}{1-z} + g_j(z). \tag{7}
$$

We define $C(z,u) = \sum_{j \geq 1} C_j(z) u^j$ and $g(z,u) = \sum_{j \geq 1} g_j(z) u^j$. Multiplying both sides of (7) by u^j and summing up for all $j \geq 1$ entails that

$$
\begin{aligned}
\frac{\partial C(z,u)}{\partial z} &= g(z,u) + \frac{C(z,u)}{1-z} + \sum_{j=1}^{\infty} \sum_{k=1}^{j-1} C_{j-k}(z) z^{k-1} u^j \\
&= g(z,u) + \frac{C(z,u)}{1-z} + \sum_{k=1}^{\infty} u^k z^{k-1} \sum_{j=k+1}^{\infty} C_{j-k}(z) u^{j-k} \\
&= g(z,u) + \frac{C(z,u)}{1-z} + \frac{u}{1-uz} C(z,u).
\end{aligned}
$$

Thus, we should solve the following differential equation with initial condition $C(0,u) = 0$:

$$
\frac{\partial C(z,u)}{\partial z} = g(z,u) + \left(\frac{1}{1-z} + \frac{u}{1-uz} \right) C(z,u). \tag{8}
$$

The homogeneous solution of the above equation is $1/((1-z)(1-uz))$. Using this homogeneous solution, one can easily obtain the solution of the partial differential equation (8) as follows

$$C(z,u) = \frac{1}{(1-z)(1-uz)}\left(\int (1-z)(1-uz)g(z,u)dz + K\right). \tag{9}$$

Here, K is a constant which should be determined according to the initial condition $C(0,u)=0$. Now, if we introduce

$$T(z,u) = \sum_{n=1}^{\infty} z^n \sum_{j=1}^{n} t_{n,j}u^j,$$
$$T_{n_0}(z,u) = \sum_{n=1}^{n_0} z^n \sum_{j=1}^{n} t_{n,j}u^j, \quad B_{n_0}(z,u) = \sum_{n=1}^{n_0} z^n \sum_{j=1}^{n} b_{n,j}u^j,$$

we obtain the following expression for $g(z,u)$ in terms of the toll generating function $T(z,u)$ and the polynomials $T_{n_0}(z,u)$ and $B_{n_0}(z,u)$:

$$\begin{aligned} g(z,u) &= \frac{\partial}{\partial z}\left(B_{n_0}(z,u) + T(z,u) - T_{n_0}(z,u)\right) + \frac{u(uz)^{n_0}}{1-uz}B_{n_0-1}(1/u,u) \\ &- \frac{u}{1-uz}B_{n_0-1}(z,u) - \frac{1}{1-z}\left(B_{n_0-1}(z,u) - z^{n_0}B_{n_0-1}(1,u)\right). \end{aligned}$$

As a consequence, the solution of the differential equation (8) is

$$\begin{aligned} C(z,u) &= \frac{\int_0^z (1-y)(1-uy)\frac{\partial}{\partial y}T(y,u)\,dy}{(1-z)(1-uz)} + B_{n_0}(z,u) - T_{n_0}(z,u) \\ &+ \frac{\int_0^z (1+u-2uy)(y^{n_0}\sum_{j=1}^{n_0} b_{n_0,j}u^j - T_{n_0}(y,u))\,dy}{(1-z)(1-uz)} \\ &+ \frac{B_{n_0-1}(1,u)\frac{z^{n_0+1}}{n_0+2}\left((1-uz)+\frac{1}{n_0+1}\right)}{(1-z)(1-uz)} \\ &+ \frac{B_{n_0-1}(1/u,u)\frac{(uz)^{n_0+1}}{n_0+2}\left((1-z)+\frac{1}{n_0+1}\right)}{(1-z)(1-uz)} \\ &= \frac{\int_0^z (1-y)(1-uy)\frac{\partial}{\partial y}T(y,u)\,dy}{(1-z)(1-uz)} + C_{n_0}(z,u) \\ &= C_T(z,u) + C_{n_0}(z,u), \end{aligned}$$

where $C_{n_0}(z,u)$ represents the part of $C(z,u)$ that depends on n_0 and

$$C_T(z,u) = \frac{\int_0^z (1-y)(1-uy)\frac{\partial}{\partial y}T(y,u)\,dy}{(1-z)(1-uz)},$$

a part which is independent of the cutoff value and the policy for small subfiles.

Since we assume $t_{n,j} = \alpha n + \beta + \gamma/(n-1)$ (see Section 4), then we get

$$
\begin{aligned}
C_T(z,u) = & \frac{(-3\alpha + \alpha u - \beta u + \beta)}{(1-u)^2} \left(\frac{1}{1-z} - \frac{u}{1-uz} \right) \log \frac{1}{1-uz} \\
+ & \frac{(3\alpha u - \alpha - \beta u + \beta)u}{(1-u)^2} \left(\frac{1}{1-z} - \frac{u}{1-uz} \right) \log \frac{1}{1-z} \\
+ & \frac{(\alpha-\beta)u}{(1-u)(1-z)} + \frac{2\alpha u}{(1-u)(1-z)^2} - \frac{(\alpha-\beta)u}{(1-u)(1-uz)} \\
- & \frac{2\alpha u}{(1-u)(1-uz)^2} + \left(\frac{\gamma u(1+u-2uz)}{6(1-u)} - \frac{\gamma(1-u)}{6(1-z)} \right) \log \frac{1}{1-uz} \\
- & \left(\frac{\gamma(1+u-2uz)}{6(1-u)} - \frac{\gamma(1-u)}{6(1-uz)} \right) \log \frac{1}{1-z} \\
- & \frac{\gamma(1+33u)}{36(1-u)(1-uz)} + \frac{\gamma(1+u-8uz)}{36} + \frac{\gamma u(u+33)}{36(1-u)(1-z)}.
\end{aligned}
$$

Moreover, if $n > n_0$ then we have $C_{n_0}(z,u) = p(z,u)/((1-z)(1-uz))$, with

$$
p(z,u) = \sum_{k=2}^{n_0+2} \sum_{i=1}^{n_0} p_{k,i} z^k u^i
$$

a polynomial in z and u that has neither $(1-z)$ nor $(1-uz)$ as a factor. Since $[z^n u^j](1-z)^{-1}(1-uz)^{-1} = [\![j \le n]\!]$ (with $[\![P]\!] = 1$ if the predicate P is true and 0 otherwise), then

$$
[z^n u^j] C_{n_0}(z,u) = \sum_{k=2}^{n_0+2} \sum_{i=1}^{n_0} p_{k,i} [\![j - i \le n - k]\!].
$$

Therefore, if $n_0 + 1 \le j \le n - n_0$ and $n \ge 2n_0 + 1$, then for $2 \le k \le n_0 + 2$ and $1 \le i \le n_0$, we have $j - i \le n - k$, and so we find

$$
r_{n_0} := [z^n u^j] C_{n_0}(z,u) = C_{n_0}(1,1), \qquad n_0 < j \le n - n_0, \quad n \ge 2n_0 + 1.
$$

If we do the calculations, we obtain for $n_0 + 1 \le j \le n - n_0$ and $n \ge 2n_0 + 1$:

$$
\begin{aligned}
r_{n_0} = & \frac{\sum_{i=1}^{n_0} \sum_{k=i}^{n_0} b_{k,i}}{(n_0+1)(n_0+2)} + \frac{\sum_{k=1}^{n_0-1} \sum_{i=1}^{n_0-k} b_{i+k-1,i}}{(n_0+1)(n_0+2)} \\
& + \frac{\sum_{i=1}^{n_0} b_{n_0,i}}{(n_0+1)(n_0+2)} - \frac{(n_0-1)(17n_0^2 + 32n_0 + 12)}{18(n_0+2)(n_0+1)n_0} \gamma \\
& + \left(6H_{n_0+2} - 2n_0 - 10 + \frac{2}{n_0+2} \right) \alpha \\
& + \left(4 - 2H_{n_0+2} - \frac{2}{n_0+2} \right) \beta.
\end{aligned} \tag{10}
$$

Equation (10) covers most of the interesting values of j, in particular $j = n/2$. The fact that $[z^n u^j]C_{n_0}(z,u)$ does not depend on j for a wide range of values of j and given a sufficiently large value of n, has an easy intuitive explanation: after the $\Theta(\log n)$ passes needed to reduce the size of the subfile from n to $m \le n_0$ the rank of the sought element relative to the subfile will be any of $1, \ldots, m$ with identical probability, no matter what the initial value of j is, apart from the extreme cases $j \le n_0$ or $j > n - n_0$.

4 Toll and cutoff functions

Up to now, we have investigated the average total cost of quickfind under a general framework for which no particular values of the toll function $t_{n,j}$ nor the cutoff function $b_{n,j}$ are given.

Comparisons	$n-1$
Passes	1
Exchanges (basic)	$n/6 + 1/3 + 2/(3(n-1))$
Exchanges (median-of-3)	$n/5 + 1/5 + 3/(5(n-1))$

Table 1: Toll functions.

In Table 1 we find the values of $t_{n,j}$ for three important measures: comparisons, exchanges and passes. None of these depends on j and for our subsequent development we shall assume that

$$t_{n,j} = \alpha n + \beta + \gamma/(n-1).$$

	Comparisons	Exchanges	Data movements
Insertion	$n(n-1)/4$		$(n+4)(n-1)/4$
Selection	$jn - j(j+1)/2$	j	

Table 2: Cutoff functions ($j \le \lceil n/2 \rceil$).

On the other hand, Table 2 lists the average number of comparisons, exchanges, and single data movements made by insertion sort and selection sort. The analysis of these basic algorithms can be found elsewhere; see for instance [6, 7, 9, 10].

Since we should actually use a slight modification of selection sort, we briefly justify the values given in Table 2. For each i, $1 \le i \le j$, the algorithm has already located and sorted the smallest $i-1$ elements in the subfile. At the i-th pass, it has to locate the minimum among the remaining $n-i+1$ elements (with $n-i$ comparisons) and bring it to its correct position, performing a redundant exchange if the sought minimum were already in its correct position. This is what selection sort does, but now the algorithm terminates when the j-th minimum in the subfile is located and brought to its correct position. Also, if $j > \lceil n/2 \rceil$, we should locate the maximum, then the second maximum, etc.

Therefore, the number of exchanges made by this algorithm is j and the number of comparisons is

$$\sum_{1 \le i \le j} (n-i) = nj - \frac{j(j+1)}{2}.$$

Guided by these two algorithms, but in order to facilitate the analysis of quickfind with alternative small subfiles algorithms, we shall work with cutoff functions of the form

$$b_{n,j} = K_1 n^2 + K_2 n + K_3 j^2 + K_4 j + K_5 jn + K_6 + K_7 jH_n + K_8 H_n,$$

where the inclusion of the two last terms is required, for instance, in the analysis of quickfind with heapselect for small subfiles. In this extended abstract, we will set $K_7 = K_8 = 0$, though.

Last but not least, we conclude this section with the simplified model of costs that we will use in the sequel. In this model, the unit of measurement is a single comparison between any two elements. We denote by ξ the cost of an exchange of elements, and by ϕ the cost of a pass (this cost includes the selection of the pivot and the cost of additional bookkeeping). Typically, $\xi \approx 3$ and $\phi \approx 2\xi \approx 6$; these values agree with the experimental values given for usual implementations and also with the values obtained from more delicate and precise cost models [6, 7, 9]. We also assume that a single data movement has cost $\xi/3$; in a real setting it would be a bit more, since exchanges use some intermediate register rather than three data movements.

Under these assumptions and measuring the total cost as the sum of the costs of comparisons, exchanges and passes, we have

$$\alpha = 1 + \xi/6, \quad \beta = \xi/3 + \phi - 1, \quad \text{and} \quad \gamma = 2\xi/3.$$

The bookkeeping costs in the outermost loop of insertion and selection sort are assimilable to "passes", but we shall introduce suitable constants ϕ_{ins} and ϕ_{sel} to account for these costs (usually, ϕ_{ins} and ϕ_{sel} are $\approx 3/2$).

In the case of insertion sort we have $n-1$ iterations through the main loop and hence

$$K_1 = \frac{1}{4}(1+\xi/3), K_2 = \frac{1}{4}(-1+\xi) + \phi_{\text{ins}}, K_6 = -\xi/3 - \phi_{\text{ins}},$$
$$K_3 = K_4 = K_5 = 0,$$

while in the case of selection sort, we perform the outermost loop j times and we have thus

$$K_3 = -\frac{1}{2}, K_4 = -\frac{1}{2} + \xi + \phi_{\text{sel}}, K_5 = 1,$$
$$K_1 = K_2 = K_6 = 0.$$

5 Optimal cutoff bounds

We are now ready to tie the general framework of Section 3 with the particular toll and cutoff functions of Section 4, to provide the optimal values of n_0. We have used Maple to instantiate (10) with the proper values of the toll and cutoff functions of Section 4, and to help with the tedious calculations[1].

Since r_{n_0} is the contribution originated by the term that depends on n_0, r_0 should be 0, since in this case we have the basic quickfind algorithm. When r_{n_0} is negative, then cutting off the algorithm (when we have files of size n_0 or less) improves its efficiency. Therefore, the optimal value of n_0 is the one that minimizes r_{n_0}.

In these calculations we have considered $\xi = 3, \phi = 6, \phi_{\text{ins}} = 3/2$, and $\phi_{\text{sel}} = 3/2$. For the case of insertion sort, we obtain

n_0	0	1	2	3	4	5	6
r_{n_0}	1.0	-2.5000	-5.0000	-6.5166	-7.3277	-7.5182	-7.1277

n_0	7	8	9	10	11	12
r_{n_0}	-6.1787	-4.6850	-2.6562	-0.9902	2.9817	6.5823

Table 3: Values of r_{n_0} for $0 \leq n_0 \leq 12$, for insertion sort.

From the table we can see that the optimal value of n_0 is 5, while quickfind with cutoff has better performance than the standard algorithm only up to $n_0 = 10$. For the case of selection sort, we obtain

From this table we can see that the optimal value of n_0 is 6, and that switching to selection sort improves the performance of the standard algorithm only up to $n_0 = 11$.

The variation of r_{n_0} as a function of the performance constants is small for reasonable ranges of variation of ξ, ϕ, ϕ_{ins} and ϕ_{sel}, which indicates that setting n_0 in the range of $n_0 = 5$ to $n_0 = 10$ provides a tuned but portable implementation with good performance. For example, if we consider $\xi = 1.5$ and $\phi = 4.5$, then the optimal value of n_0 is is 4 for insertion sort and 5 for selection sort, while if we consider $\xi = 2.5$ and $\phi = 7.5$ then the optimal values of n_0 are respectively 5 and 7.

[1]The Maple program is available from request from the contact author (conrado@lsi.upc.es).

n_0	0	1	2	3	4	5	6
r_{n_0}	0.	-1.0000	-2.8333	-4.1166	-4.9944	-5.4944	-5.6277

n_0	7	8	9	10	11	12
r_{n_0}	-5.4009	-4.8183	-3.8835	-2.5990	-0.9669	1.0108

Table 4: Values of r_{n_0} for $0 \leq n_0 \leq 12$, for selection sort.

On the other hand, experimental determination of the performance constants (or even better, of the constants α, β, γ and K_i) together with (10) would yield a precise estimation of the optimal cutoff value n_0 for any particular implementation and environment under which the algorithm is run.

6 Conclusions

In this paper we have investigated the variants of the Quickfind algorithm for the selection of the j-th element out of n, when recursion stops for subfiles whose size is below a predefined threshold, and a simpler algorithm is run instead. This analysis requires considering the expected total cost of the algorithm (not just the expected number of comparisons), a cost which combines the number of passes, comparisons and exchanges made by the algorithm.

We have considered two policies for the small subfiles: insertion sort and an adaptation of selection sort, but our analysis is general enough to be easily applied to alternative policies. We have obtained the average cost for each of these variants and compared them with the costs of standard variants which do not use cutoff. We have also given the best explicit cutoff bound for each of the variants.

In the final version of this extended abstract we will report our results for the median-of-three variant of quickfind.

References

[1] Anderson, D., and Brown, R. Combinatorial aspects of C.A.R. Hoare's Find algorithm. *Australasian Journal of Combinatorics 5* (1992), 109–119.

[2] Hoare, C. Find (Algorithm 65). *Communications of the ACM 4* (1961), 321–322.

[3] Hoare, C. Quicksort. *Computer Journal 5* (1962), 10–15.

[4] Kirschenhofer, P., Prodinger, H., and Martínez, C. Analysis of Hoare's Find algorithm with median-of-three partition. *Random Structures & Algorithms 10* (1997), 143–156.

[5] Knuth, D. Mathematical analysis of algorithms. In *Information Processing '71, Proc. of the 1971 IFIP Congress* (Amsterdam, 1972), North-Holland, pp. 19–27.

[6] Knuth, D. *The Art of Computer Programming: Sorting and Searching*, 2nd ed., vol. 3. Addison-Wesley, Reading, Mass., 1998.

[7] Sedgewick, R. The analysis of quicksort programs. *Acta Informatica 7* (1976), 327–355.

[8] Sedgewick, R. Implementing quicksort programs. *Communications of the ACM 21* (1978), 847–856.

[9] Sedgewick, R. *Quicksort.* Garland, New York, 1978.

[10] Sedgewick, R., and Flajolet, P. *An Introduction to the Analysis of Algorithms.* Addison-Wesley, Reading, Mass., 1996.