Algorithms Seminar 2002–2004,
F. Chyzak (ed.), INRIA, (2005), pp. 101–104.

Available online at the URL
http://algo.inria.fr/seminars/.

Forty Years of 'Quicksort' and 'Quickselect': a Personal View

Conrado Martínez
Universitat Politècnica de Cataluny (Spain)

October 6, 2003

Summary by Marianne Durand

Abstract

The algorithms 'quicksort' [3] and 'quickselect' [2], invented by Hoare, are simple and elegant solutions to two basic problems: sorting and selection. They are widely studied, and we focus here on the average cost of these algorithms, depending on the choice of the sample. We also present a partial sorting algorithm named 'partial quicksort'.

1. 'Quicksort' and 'Quickselect'

The sorting algorithm 'quicksort' is based on the divide-and-conquer principle. The algorithm proceeds as follows. First a pivot is chosen, with a specified strategy. Then all the elements of the array but the pivot are compared to the pivot. The elements smaller than the pivot are stored before the pivot, and the elements larger after the pivot. These two sub-arrays are then recursively sorted.

We denote by C_n the average number of comparisons done to sort an array of size n, and $\pi_{n,k}$ the probability that the kth element is the chosen pivot in an array of size n. The recursive design of the algorithm is translated into a recurrence satisfied by the cost C_n:

$$C_n = n - 1 + t_n + \sum_{k=1}^{n} \pi_{n,k}(C_{k-1} + C_{n-k}). \tag{1}$$

The value t_n denote the cost, in terms of comparisons, of the choice of the pivot, that may depend on n.

The selection algorithm 'quickselect' is based on the same principles. To select the mth element out of an array of size n, first a pivot is chosen, then the array is partitioned into two sub-arrays, and the mth element is then recursively selected into the appropriate sub-array.

The average cost of selecting the mth element in an array of size n, in terms of comparisons, using this algorithm is denoted by $Q_{n,m}$, which satisfies the recurrence

$$Q_{n,m} = n - 1 + t_n + \sum_{k=1}^{m-1} \pi_{n,k} Q_{n-k,m-k} + \sum_{k=m+1}^{n} \pi_{n,k} Q_{k-1,m}. \tag{2}$$

In the particular case of the standard variant, where the pivot is chosen with a uniform probability ($\pi_{n,k} = 1/n$), those recurrences are solved, and lead to the theorem:

Theorem 1. *The average number of comparisons to sort n elements using the standard 'quicksort'* [3] *is*

$$C_n = 2(n+1)H_n - 4n \sim 2n \log n, \tag{3}$$

where H_k is the kth harmonic number. The average number of comparisons $Q_{n,m}$ to select the mth element out of n elements using the standard 'quickselect' [5] *is*

$$Q_{n,m} = 2((n+1)H_n - (n+3-m)H_{n+1-m} - (m+2)H_m + n + 3) = \Theta(n). \tag{4}$$

The maximum of the function $Q_{n,\alpha n}$ is located at $\alpha = 0.5$ and is worth $(2 + 2\log 2)n$.

2. Different Sampling Strategies

To improve the cost of these two algorithms, a first idea is to use a different algorithm for small subfiles (for example insertion sort), and a second idea is to use samples to select better pivots, and reduce the probability of uneven partitions which lead to quadratic worst case.

2.1. **Median of 3.** A simple strategy to select a pivot, due to Singleton [11], is *median of 3*. The pivot is chosen as the median of a sample of size 3, selected with uniform probability. The distribution of the pivot is now characterized by $\pi_{n,k} = \frac{(k-1)(n-k)}{\binom{n}{3}}$.

The average number of comparisons in this case is equal to ([10])

$$C_n = \frac{12}{7} n \log n + O(n),$$

which is roughly 14% less than standard 'quicksort'.

The median-of-3 strategy can also be applied to the algorithm 'quickselect'. Kirshenhofer, Martinez, and Prodinger studied this variant in [4]. By using bivariate generating functions and technical exact computation, they found that the average number of comparisons is

$$Q_{n,m} = 2n + \frac{72}{35} H_n - \frac{156}{35} H_m - \frac{156}{35} H_{n+1-m} + 3m - \frac{(m-1)(m-2)}{n} + O(1).$$

In the particular case where $m = \lceil n/2 \rceil$, the average number of comparisons is $\frac{11}{4} n + o(n)$, which is 18% less than in the standard case.

2.2. **Optimal sampling.** The median-of-3 strategy can be generalized to the median-of-$2t+1$ strategy for any integer t. Martínez and Roura in [6] consider the case where the size of the sample is $s = 2t + 1$, with $t = t(n)$ depending on n. Traditional techniques to solve recurrences cannot be used here. Their approach is to make an extensive use of the continuous master theorem of Roura [9]. This theorem states that if the sequence F_n satisfies the recursive equation

$$\left\{ \begin{array}{lcl} F_n & = & b_n \quad \text{if} n < N \\ F_n & = & t_n \sum_k w_{n,k} F_k \end{array} \right. , \tag{5}$$

with appropriate growth conditions on the $w_{n,k}$, then the asymptotic behavior of F_n is known. It depends on the growth of the toll function t_n and of the coefficients $w_{n,k}$, and is equivalent to $t_n \log^k n (\log\log n)^i$ or $n^\alpha \log^k n$, where all the coefficients involved are known. This theorem is the appropriate tool to deal with the recurrences satisfied by the cost of 'quicksort median of $2t+1$'.

The complexity considered in [6] is the *total cost*. The total cost is function of the number of comparisons and of the number of exchanges, and is defined by $\#comparisons + \zeta \#exchanges$, where ζ is usually considered to be around 4. We state the following theorem on the total cost of the algorithms 'quicksort' and 'quickselect'.

Theorem 2. *If we use samples of size s, with $s = o(n)$ and $s = \omega(1)$, then the average total cost of 'quicksort' is*

$$C_n = (1 + \zeta/4) n \log_2 n + o(n \log n). \tag{6}$$

The average total cost of 'quickselect', with the same sample strategy, to find an element of given random rank is

$$Q_n = 2(1 + \zeta/4)n + o(n). \tag{7}$$

The optimal sample size s^*, that minimizes the total cost of 'quicksort' and 'quickselect', satisfies $s^* = O(\sqrt{n})$, and depends on the number of comparisons done to select the pivot.

The conclusion is that if exchanges are expensive, we should use fixed-size samples and pick the median.

Many other strategies of pivot selection are available. For example 'median-3-3 quicksort', where the pivot is the median of three medians of three samples, each sample of size three [1]. This leads to all the strategies where the pivot is a median of other preselected medians, each issued of a selection strategy of the same type. The idea to keep in mind is that the gain obtained by a better pivot strategy should always be larger than the additional cost of the choice of the pivot.

2.3. **Adaptive sampling.** For the 'quicksort' algorithm, the best pivot is the median of the array. This is not obviously the best choice for 'quickselect', for example if m is small or close to n. The idea of Martínez, Panario, and Viola in [8] is to choose a pivot with relative rank in the sample close to $\alpha = m/n$.

3. Partial Sort

The partial sort problem is, given an array of size n, sort the m smallest elements. The algorithm 'quickselsort' answers this question. It selects the mth element by 'quickselect', and then applies 'quicksort' to the $m-1$ elements to its left. The cost of this algorithm is $\Theta(n + m \log m)$. Another way is the *heapsort-based partial sort*, that builds a heap and extracts m times the minimum. Its cost is also $\Theta(n + m \log m)$.

A solution, given by Martínez in [7], is 'partial quicksort'. This algorithm uses the principles of 'quicksort', and proceeds as follows. First find a pivot (with any strategy) and then recursively apply 'partial quicksort' to the sub-arrays concerned. More precisely, if the pivot is smaller than m, sort the left sub-array, and apply 'partial quicksort' to the right sub-array. If the pivot is greater than m, then apply 'partial quicksort' to the left sub-array.

The average number of comparisons $P_{n,m}$ needed to sort the m smallest elements in an array of size n satisfy the recurrence

$$P_{n,m} = n - 1 + t + \sum_{k=m+1}^{n} \pi_{n,k} P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \left(P_{k-1,k-1} + P_{n-k,m-k} \right).$$

In this recurrence, t is the number of comparisons done to choose the pivot, and k represents the position of the pivot, chosen with probability $\pi_{n,k}$. When k is greater than m, the algorithm sorts the m smallest elements in the left sub-array, that has size $k-1$. In the other case, when k is smaller than m, the algorithm sorts the entire left sub-array, and the $m-k$ smallest elements of the right sub-array, that has size $n-k$. We recognize that $P_{n,n}$ is the average cost of the algorithm 'quicksort'.

In the standard case, when $\pi_{n,k} = 1/n$, this recurrence is solved exactly, and we get that the average number of comparisons done by 'partial quicksort' is

$$P_{n,m} = 2n + 2(n+1)H_n - 2(n+3-m)H_{n+1-m} - 6m + 6. \tag{8}$$

'Partial quicksort' makes $2m - 4H_m + 2$ comparisons and $m/3 - 5H - m/6 + 1/2$ exchanges less than 'quickselsort'.

Bibliography

[1] Bentley (Jon L.) and McIlroy (Douglas). – Engineering a sort function. *Software—Practice and Experience*, vol. 23, n° 11, 1993, pp. 1249–1265.

[2] Hoare (Charles A. R.). – Find. *Communications of the ACM*, vol. 4, n° 7, 1961, pp. 321–322.

[3] Hoare (Charles A. R.). – Quicksort. *The Computer Journal*, vol. 5, n° 1, 1962, pp. 10–15.

[4] Kirschenhofer (P.), Prodinger (H.), and Martínez (C.). – Analysis of Hoare's FIND algorithm with median-of-three partition. *Random Structures & Algorithms*, vol. 10, n° 1–2, 1997, pp. 143–156.

[5] Knuth (Donald E.). – *The Art of Computer Programming*. – Addison-Wesley, 1998, seconde edition, vol. 3: Sorting and Searching.

[6] Martínez (C.) and Roura (S.). – Optimal sampling strategies in quicksort and quickselect. *SIAM Journal on Computing*, vol. 31, n° 3, 2001, pp. 683–705.

[7] Martinez (Conrado). – Partial quicksort. In *Proceedings of the First ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. – 2004.

[8] Martínez (Conrado), Panario (Daniel), and Viola (Alfredo). – Adaptive sampling for quickselect. In *Proceedings of the 15thACM-SIAM Symp. on Discrete Algorithms (SODA)*. – 2004.

[9] Roura (Salvador). – An improved master theorem for divide-and-conquer recurrences. In *Automata, languages and programming (Bologna, 1997)*. pp. 449–459. – Springer, Berlin, 1997.

[10] Sedgewick (R.). – The analysis of quicksort programs. *Acta Informatica*, vol. 7, 1977, pp. 327–355.

[11] Singleton (Richard C.). – Algorithm 347: an efficient algorithm for sorting with minimal storage [m1]. *Communications of the ACM*, vol. 12, 1969, pp. 185 – 186.