In Proceedings of the 28th Annual Allerton Conference on Communication, Control, and Computing, October 1990, pp. 62-71

Optimal Sampling Strategies for Quicksort

C. C. McGeoch*
DIMACS Center
Rutgers University

J. D. Tygar †
School of Computer Science
Carnegie-Mellon University

Abstract

A well-known variant of Quicksort samples from the subarray at each recursive stage and uses the sample median as the partition element. Three families of *general sampling strategies*, which allow sample size to vary as a function of subarray size, are considered. The total cost of sorting comparisons plus median-selection comparisons is analyzed. A strategy which takes a sample of size $\theta(\sqrt{n})$ for subarray size n is shown to be optimal over a large class of strategies. The square-root strategy has $O(n^{1.5})$ worst-case cost. The true optimal strategy is found computationally for $n < 10,000$.

1 Introduction

The basic operation of Quicksort partitions an array $X = x_1, x_2, \ldots, x_n$ around a partition element $p = x_i$ so that elements with value less than p are to its left in the array and larger elements are to its right; Quicksort then recurs on the two subarrays on either side of p. A well-known generalization (see [8], [14], or [13]) takes a sample of fixed size (usually 3) at each recursive stage and uses the sample median as the partition element; we call this a *fixed-sample (FS) strategy.*

We introduce a class of Quicksort algorithms for which sample size can vary as a function of subarray size at each recursive stage. A selection strategy S corresponds to a *sampling function* $\sigma(n)$ which takes subarray size n and returns a sample size s. At each recursive stage, the *partitioning cost* is the number of comparisons required to partition a subarray of size n and the *selection cost* is the expected number of comparisons required to find the median s items. Intuition suggests that large sample sizes will decrease partitioning cost but increase selection cost: we analyze the *total cost*, of partitioning

*Permanently at: Dept. of Mathematics and Computer Science, Amherst College, Amherst MA 01002.

†Supported by National Science Foundation Presidential Young Investigators Grant CCR-88-508087. The views and conclusions in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U. S. Government.

plus selection, summed over all recursive stages. This cost model and the generalization to varying sample sizes are new to the analysis of Quicksort.

We present the following results:

- A *two-tier (TT) strategy* takes a sample of size $s = \sigma(n)$ at the first stage only; at lower levels of recursion the sample is of fixed size t. We derive an exact formula for the optimal sampling function $\sigma(n)$ for any t. For any problem size N the optimal TT strategy has lower total cost than any FS strategy.

- A *square-root (SQ) strategy* takes a sample of size $\theta(\sqrt{n})$ at all recursive stages. We describe a square-root strategy which has lower total cost than any TT strategy.

- Any SQ strategy requires $O(n^{1.5})$ total comparisons in the worst case. This is the only Quicksort-based algorithm that we know of with subquadratic worst-case cost.

- For given problem size we can compute exactly the optimal strategy; we compare this strategy to standard fixed-sample strategies for $N < 10000$.

The original Quicksort algorithm, proposed by Hoare [5, 6] selected the leftmost subarray element at each recursive stage. Hoare noted that a randomly-selected partition element would ensure expected-case behavior and suggested that larger samples might improve performance. Singleton [16] and Sedgewick [14, 13] considered FS strategies and recommended *median-of-3* Quicksort where sample size t equals 3 at all recursive stages. The analyses were not made precise because selection cost was not an explicit part of the model; rather it was considered part of the overhead of each recursive call.

Many strategies have been proposed to improve performance of Quicksort when something is known about the initial input permutation, for example when the elements are "mostly sorted." For comparative studies of such variations, see [1], [9] and [10].

We require a median-selection algorithm as a subroutine. Hoare [5] presented a selection algorithm called FIND which behaves similarly to Quicksort. Let H_m denotes the m^{th} harmonic number. Knuth ([8] Exercise 5.2.2-32) showed that the expected number of comparisons to select the median of s elements (with $s = 2m - 1$) by this algorithm is

$$FIND(m) = 4m(H_{2m-1} - H_m) + 4m - 4H_m + \frac{4}{3}.$$

The last term disappears when $m = 1$. Asymptotically this function has value near $3.39s$.

Floyd and Rivest [2, 3] presented a selection algorithm requiring $3s/2 + O(\sqrt{s})$ comparisons to find the median in the expected case. Both FIND and SELECT can require a quadratic number of comparisons in the worst case: Schönhage, Paterson, and Pippenger [15] describe an algorithm that finds the median of s elements using $3s+o(s)$ comparisons in the worst case.

2 Analyses of Selection Strategies

We assume that the keys to be sorted are distinct, and that all initial input permutations are equally likely; for analysis purposes, this is equivalent to taking a random sample of the elements at each level. We assume throughout that the sample size is found by rounding the sampling function value to the nearest odd integer. We follow standard analyses in assuming that a sample is taken with replacement when $t > n$ and that partitioning cost at a single stage is equal to $n + 1$.

It will be convenient to distinguish between N, the input problem size, and n, the size of a subarray somewhere during the recursion. Thus we have $n = N$ only at the first stage of recursion.

Suppose that $\sigma(n)$ returns odd positive integers and let $h(n) = (\sigma(n) - 1)/2$. Let $f(s)$ denote the expected cost of selecting the median of s items. Then the total expected cost of Quicksort with sampling strategy $\sigma(n)$ is given by the following recurrence.

$$C_\sigma(n) = n + 1 + f(\sigma(n)) + 2\sum_{r=1}^{n} \frac{\binom{n-r}{h(n)}\binom{r-1}{h(n)}}{\binom{n}{\sigma(n)}} [C_\sigma(r-1)]$$

The summation term gives the probability that the element with rank r is the sample median: equivalently, that a sample of size $\sigma(n)$ is chosen so that $h(n)$ items have rank below r and $h(n)$ items have rank above r.

We would like to derive a closed form for arbitrary $\sigma(n)$ and find the $\sigma(n)$ that minimizes $C_{\sigma(n)}$; clearly this is quite difficult. Instead, we will find optimal functions for certain classes of strategies. We consider the following classes.

1. In *fixed-sample (FS) strategies* the sample size $\sigma(n)$ equals a constant t for all $n \in 1..N$. (When $n < t$ a sample of size t is drawn with replacement from the subarray.) Total costs for these strategies can be derived from Sedgewick's [13], [14] analyses.

2. In *two-tier (TT) strategies* the sample size is determined by a function of the form

$$\sigma_t(n) = \begin{cases} \gamma(t,n) & \text{if } n = N \\ t & \text{if } n < N \end{cases}$$

   We derive the optimal formula $\gamma^*(t,n)$ for any t; it turns out that $\gamma^*(t,n) = \Theta(\sqrt{n})$ for any fixed t.

   When $\gamma^*(t,n)$ is applied, the "globally optimal" choice of t for given problem size N is denoted t^*. We show that the globally optimal TT strategy has lower cost than any FS strategy.

3. *Square-root (SQ) strategies* have sampling functions of the form $\sigma(n) = \Theta(\sqrt{n})$. We prove that the SQ strategy which applies

$$\sigma(n) = \gamma^*(t^*,n) \qquad \text{for} \quad 1 \le n \le N$$

has lower total cost than the globally optimal TT strategy.

We also show that any SQ strategy has sub-quadratic total cost in the worst case.

2.1 Optimal TT Strategies

We first derive the optimal two tier strategy $\gamma^*(1,n)$ for $t = 1$. Sedgewick's [14] [13] analyses imply that, with $t = 1$, total expected partition cost for a problem of size n is $2(n+1)(H_{n+1} - 1)$ Selection cost is clearly zero for a one-element sample. Therefore the expected cost of a TT strategy with $t = 1$ is

$$TT_1(n) = n + 1 + f(2h+1) + \frac{2}{\binom{n}{s}} \sum_{r=1}^{n} \binom{n-r}{h} \binom{r-1}{h} 2r(H_r - 1).$$

Considering the summation term alone, we have

$$\sum_{r=1}^{n} \binom{n-r}{h} \binom{r-1}{h} 2rH_r - \sum_{r=1}^{n} \binom{n-r}{h} \binom{r-1}{h} 2r,$$

which is equivalent to

$$2(h+1) \sum_{r=1}^{n} \binom{n-r}{h} \binom{r}{h+1} H_r - 2(h+1) \sum_{r=1}^{n} \binom{n-r}{h} \binom{r}{h+1}.$$

Using identities found for example in [4], we obtain

$$2(h+1) \binom{n+1}{2h+2} \left(H_{n+1} - H_{2h+2} + H_{h+1} - 1\right),$$

and the complete formula for expected total cost is therefore

$$TT_1(n) = n + 1 + f(2h+1) + 2(n+1)\left(H_{n+1} - H_{2h+2} + H_{h+1} - 1\right).$$

Our goal is to find the value of h (and therefore of s) that minimizes $TT_1(n)$. Assume that a linear-expected-time selection algorithm is used, and replace $f(2h+1)$ by the first-term approximation $a(2h+1)$ for appropriate constant a. Replacing each harmonic number by the first three terms of the approximation $H_n \approx \ln n + \delta + 1/2n - O(1/n^2)$ (where $\delta \approx .577216$ is Euler's constant), we obtain

$$TT_1(n) \approx n + 1 + a(2h+1) + 2(n+1)\left(\ln(n+1) - \ln 2 + \delta + \frac{1}{2(n+1)} + \frac{1}{4(h+1)}\right).$$

This is minimized when

$$h^* = \sqrt{\frac{(n+1)}{4a}} - 1 \qquad \text{or} \qquad s^* = \gamma^*(1,n) = \sqrt{(n+1)/a} - 1.$$

Because of the approximation we have only found the first term of the optimal sampling function. This function must be rounded to the nearest odd integer to obtain a sample size: therefore the approximation is sufficient to give the true optimal sample size

asymptotically, since with $\gamma^*(1,n) = \theta(\sqrt{n})$ the approximation is accurate to within an $O(1/n)$ term.

This analysis can be generalized to show that the optimal strategy for any t is a square-root function of n. Let $s = 2h+1$ as before and, similarly, let $t = 2k+1$. We have total partitioning cost

$$P_t(n) = \frac{(n+1)H_{n+1}}{H_{2k+2} - H_{k+1}} + O(n),$$

and total expected selection cost equal to $\Theta(nt)$ (see [14], [13]). Then

$$TT_t(n) = n + 1 + f(2h+1) + 2\sum_{r=1}^{n}\left[\frac{\binom{r-1}{h}\binom{n-r}{h}}{\binom{n}{2h+1}}\frac{rH_r}{H_{2k+2} - H_{k+1}} + \Theta(kr)\right]$$

which has closed form

$$TT_t(n) = n + 1 + f(2h+1) + (n+1)\left[\frac{H_{n+1} - H_{2h+2} + H_{h+1}}{H_{2k+2} - H_{k+1}} + \Theta(k)\right].$$

Optimal sample size is given by

$$h^* = \sqrt{\frac{n+1}{8a\,(H_{2k+2} - H_{k+1})}} - 1 \qquad \text{or} \qquad s^* = \gamma^*(t,n) = \sqrt{\frac{n+1}{2a\,(H_{2k+2} - H_{k+1})}} - 1. \quad (1)$$

The sum $H_{2k+2} - H_{k+1}$ is approximately equal to $\ln 2 - 1/(4k+4) \approx .693 - 1/(2t+2)$.

For any problem size N there must be some optimal t^* that minimizes total comparison cost over all TT strategies which use $\gamma^*(t,n)$. Our computational results suggest that optimal t^* is given by a slowly increasing function of N: for now we denote the optimal sampling function by

$$\sigma_N^*(n) = \begin{cases} \gamma^*(t^*,n) & \text{if } n = N \\ t^* & \text{if } n < N \end{cases}$$

Thus the optimal strategy is a "global" strategy for problem size N rather than one that only depends upon subarray size. The total cost of sorting an array of size n by this strategy is denoted by $TT_N^*(n)$.

2.2 The Square-Root Strategy

We now consider a *square-root strategy* which applies the sampling function

$$\sigma(n) = \gamma^*(t^*,n) \qquad \text{for} \quad 1 \le n \le N.$$

The $\gamma^*(t^*,n)$ and t^* here are as derived above for the optimal TT strategy.

We can rank the following five strategies.

- Let $\widehat{FS}(N)$ denote the cost of the optimal FS strategy which applies $\sigma(n) = \hat{t}$, where $\hat{t}$ is optimal for N. Optimal $\hat{t}$ grows slowly with problem size N (see [14], [13]).

- Let $FS^*(N)$ denote the expected cost of the FS strategy that applies $\sigma(n) = t^*$, where t^* is optimal for TT strategies. Note that t^* does not necessarily equal $\hat{t}$.
- Let $TT^*_N(N)$ denote the cost of the globally optimal TT strategy for problem size N, derived in the previous section.
- Let $\widehat{TT}_N(N)$ denote the cost of the TT strategy that applies

$$\sigma(n) = \begin{cases} \gamma^*(\hat{t}, n) & \text{if } n = N \\ \hat{t} & \text{if } n < N \end{cases}$$

  where $\hat{t}$ is as defined above.
- Let $SQ(n)$ denote the cost of the strategy which applies

$$\sigma(n) = \gamma^*(t^*, n) \qquad \text{for} \quad 1 \leq n \leq N.$$

With these definitions, we have the following theorem.

Theorem: *For a problem of size N, total comparison costs are related by*

$$SQ(N) \leq TT^*_N(N) \leq \widehat{TT}_N(N) \leq \widehat{FS}(N) \leq FS^*(N).$$

Proof: Note that the problem size N is fixed beforehand and determines the globally optimal t^* and $\hat{t}$ for their respective strategy classes. The theorem claims only that the relationships hold for the *total* expected cost of sorting, not necessarily for $n < N$.

We will demonstrate the inequalities working from right to left and numbering the arguments for reference.

(1) The fourth inequality is immediate by the definition of $\hat{t}$ as optimal for fixed-sample strategies at problem size N.

(2) The third inequality is immediate by the derivation of $\gamma^*(t, n)$ as optimal for any t. When $n < N$ the two strategies are identical. When $n = N$, the two-tier strategy choses a sample $\gamma^*(\hat{t}, n)$ which is optimal.

(3) The second inequality is immediate from the definition of t^* as optimal for all TT strategies with problem size N.

(4) We shall establish the first inequality by induction on problem size N. Let R be an arbitrary value smaller than N, and consider the TT^*_R strategy that applies

$$\sigma_R(n) = \begin{cases} \gamma^*(t^*, n) & \text{if } n = R \\ t^* & \text{if } n < R \end{cases}$$

The cost of sorting an array of size R by this strategy is denoted $TT^*_R(R)$. This is different from the TT^*_N strategy above because the "top level" for this strategy is at $n = R$ rather than at $n = N$. (The t^* is the same in both, however).

The induction hypothesis is that $SQ(R) \leq TT^*_R(R)$ for all $R < N$. Let $FS^*(R)$ be the cost of applying the FS strategy (with t^*) to a problem of size R. By the optimality of $\gamma^*(t^*, n)$ we have

$$SQ(R) \leq TT^*_R(R) \leq FS^*(R).$$

Now, $TT^*_N(R) = FS^*(R)$ whenever $R < N$, since the sampling functions are identical. Therefore we have $SQ(R) \leq TT^*_N(R)$ whenever $R < N$.

Consider now the costs $SQ(N)$ and $TT^*_N(N)$. When $n = N$, the two strategies are identical since they both take samples of size $\gamma^*(t^*, N)$. Letting $C(N)$ denote partition plus selection cost when $n = N$, and letting $P(N, r)$ denoting the probability that the partition element has rank r, the cost functions for the two strategies are

$$SQ(N) = C(N) + 2\sum_{r=1}^{N} P(N, r)\, SQ(r)$$

and

$$TT^*_N(N) = C(N) + 2\sum_{r=1}^{N} P(N, r)\, TT^*_N(r).$$

Since $SQ(r) \leq TT^*_N(r)$ for all $r < N$ the theorem is proved. As a basis case, note that γ^* will return 1, easily shown to be optimal for small n, when n is small. □

Argument (4) implies the that SQ strategy is better at all recursive stages.

Corollary: $SQ(n) \leq ST^*_N(n)$ for all $1 \leq n \leq N$.

Finally, it is easy to demonstrate subquadratic worst-case performance for any square-root strategy, even when the median-selection routine has quadratic worst-case cost.

Theorem: *The worst-case total cost of any square-root strategy is* $O(n^{1.5})$.

Proof: Suppose that a selection algorithm with worst-case cost $O(s^2)$ is used. To simplify notation, assume that the sample drawn is of size at least $2b\sqrt{n} + 1$, and at most $c\sqrt{n}$ for appropriate constants b and c. Worst-case partitioning cost occurs when extremal values of every subarray are chosen as partition elements. But if a sample of size at least $2b\sqrt{n}+1$ is drawn at each level, it is impossible for elements with rank less than or equal to $b\sqrt{n}$ (or greater than $n - b\sqrt{n}$) to become sample medians. Therefore the worst-case cost for any SQ strategy is bounded by the following recurrence,

$$SQ(n) \leq n + 1 + (c\sqrt{n})^2 + SQ(b\sqrt{n}) + SQ(n - b\sqrt{n}),$$

which has closed form $O(n^{1.5})$. Note that this bound holds even if worst-case selection cost is quadratic in the sample size.

n	35	93	197	337	515	730	984	1274	1603	1968	2372
s_n	3	5	7	9	11	13	15	17	19	21	23
n	2813	3292	3808	4362	4953	5582	6248	6952	7693	8472	9289
s_n	25	27	29	31	33	35	37	39	41	43	45

Figure 1: Optimal Sample Size

3 An Exact Optimal Strategy

Given an exact formula $f(s)$ for selection cost it is possible to compute directly the optimal sample size by a dynamic-programming algorithm that finds $s = \sigma(n)$ to minimize

$$OPT(n) = n + 1 + f(s) + \sum_{r=1}^{n} \frac{\binom{r-1}{k}\binom{n-r}{k}}{\binom{n}{s_n}}[OPT(r-1) + OPT(n-r)].$$

The running time of the computation is just quadratic in the largest n examined, since it is only necessary to compare the current optimal value of s to $s+2$ at each value of n, rather than considering all possible values.

Hoare's selection algorithm FIND, although not the most efficient known, does permit an exact formula for selection cost, given in Section 1. Using this cost for $f(s)$ in the above computation of $OPT(n)$, the dynamic program produces the optimal sample sizes shown in Figure 1: for example, 5 is the optimal sample size for $93 \leq n < 197$. The computations were performed for n up to 10,000. Linear least-squares regression on a log-log scale suggests that the best power-law fit for this data is proportional to $n^{.482}$ (with residuals curving upwards); this is nearly a square-root strategy.

Table 2 compares total expected cost for the optimal strategy to fixed-sample strategies with $t = 1$ and $t = 3$. The numbers in parentheses give the ratio to the lower bound $N\lg(N) - N/\lg(N)$ (see [7]) on expected comparisons for sorting.

The fixed-sample costs $C_1(n)$ and $C_3(n)$ correspond to standard implementations found in [14] [13] and [8] (Section 5.2.2 and Exercises). These implementations are highly tuned to minimize running time in the MIX instruction set. The optimal strategy uses an inefficient general-purpose selection algorithm (Hoare's) and does not adopt some further improvements mentioned below. Nevertheless, the total comparison cost is less than either fixed strategy for N greater than 4000.

4 Remarks

Further limited experiments suggest that comparison cost is quite robust with respect to small variations in the sampling function; perhaps a hybrid strategy that choses one of a small set of sample sizes, each corresponding to a highly tuned subroutine, would give minimum execution time.

N	2000	4000	6000	8000
$C_opt(N)$	24664	53502	83879	115254
	(1.14)	(1.13)	(1.12)	(1.12)
$C_3(N)$	24420	53581	84535	116654
	(1.12)	(1.13)	(1.13)	(1.13)
$C_1(N)$	27395	60321	95339	131716
	(1.26)	(1.27)	(1.27)	(1.28)

Figure 2: Total comparison costs for three strategies.

Since all median-selection algorithms manage to partition a sample around the median, a significant reduction in comparisons could be realized by avoiding re-examining the sample during the partitioning stage. The partitioning cost at each stage would be reduced from $n-1$ to $n-s$. If the sample region is contiguous within the subarray, sentinels can be used to eliminate boundary checks.

Finally we note that a strategy that minimizes total comparisons does not necessarily minimize total running time: the number of *exchanges* of items also contributes to the leading term of running time. Experimental comparisons of fixed-sample strategies and the optimal strategy for other cost measures are described in [11] and [12].

Acknowledgements

We thank Jon Bentley who first suggested the idea of allowing sample sizes to vary as a function of array size, and Mike Langston who suggested the hybrid scheme.

References

[1] C. R. Cook and D. J. Kim. Best sorting algorithm for nearly sorted lists. *Communications of the ACM*, 23(11):620–624, November 1980.

[2] R. W. Floyd and R. L. Rivest. The algorithm select—for finding the ith smallest of n elements. *Communications of the ACM*, 18(3):173, March 1975.

[3] R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *Communications of the ACM*, 18(3):165–172, March 1975.

[4] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Welsey Publishing Company, Reading, MA, 1989.

[5] C. A. R. Hoare. Partition (algorithm 63), quicksort (algorithm 64), and find (algorithm 65). *Communications of the ACM*, 4(7):321–322, July 1961.

[6] C. A. R. Hoare. Quicksort. *Computer Journal*, 5(4):10–15, April 1962.

[7] D. E. Knuth. *The Art of Computer Programming: Volume 2, Seminumerical Algorithms.* Addison-Wesley Publishing Company, Reading, MA, 1973.

[8] D. E. Knuth. *The Art of Computer Programming: Volume 3, Sorting and Searching.* Addison-Wesley Publishing Company, Reading, MA, 1973.

[9] R. Loeser. Some performance tests of "quicksort" and "descendents.". *Communications of the ACM*, 17(3):143–152, March 1974.

[10] R. Loeser. Survey on algorithms 347, 426, and quicksort. *ACM Transactions on Mathematical Software*, 2(3):290–299, September 1976.

[11] C. C. McGeoch. *Experimental Analysis of Algorithms.* Ph.D. Dissertation, CMU-CS-87-124, Carnegie Mellon University, 1986.

[12] C. C. McGeoch. An experimental study of median-selection in quicksort. *Proceedings, 24th Allerton Conference on Communication, Control, and Computing, Univ. of Illinois*, 19–28, October 1986.

[13] R. Sedgewick. Analysis of quicksort programs. *Acta Informatica*, 7(4):327–355, 1977.

[14] R. Sedgewick. *Quicksort.* PhD thesis, Stanford University, 1975.

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

[16] R. C. Singleton. An efficient algorithm for sorting with minimal storage (algorithm 347). *Communications of the ACM*, 12(3):185–186, March 1969.