Alcom-FT Technical Report Series
ALCOMFT-TR-02-128

Asymptotic analysis of an optimized quicksort algorithm

Marianne Durand [1]

Algorithms Project, INRIA Rocquencourt, 78153 Le Chesnay, France,
marianne.durand@inria.fr

Abstract

Jon Bentley and Douglas McIlroy have implemented a fast quicksort for the C standard library in 1993. We consider here the average-case complexity in terms of number of comparisons of this algorithm, and give its asymptotic expansion up to the constant order.

Key words: quicksort, Euler equations, analysis of algorithms

In 1993, Jon Bentley and Douglas McIlroy implemented a new sorting program based on various quicksorts. Their original motivation was that the existing sorting algorithms were sometimes very slow on special inputs, and they also took notice that on average, the complexity could be improved. Their algorithm mixes three different quicksorts, the classical one, the median-of-3 quicksort, and the pseudomedian-of-9 quicksort that are explained later. The three different strategies are used on arrays of different sizes, given that the choice of the pivot can be time consuming if the sample (from which the pivot is drawn) is large.

Bentley and McIlroy carried out experiments to get an empirical complexity for this algorithm, they found that the average cost to sort an array of size n was: $f_n = 1.5783\, n \log_e n - 0.74\, n$. Our goal in this paper is to validate by theory their empirical estimates. The first asymptotic order ($O(n \log n)$) is the easiest to obtain. Here we also aim at determining the second order term that cannot be neglected, as a sorting algorithm has to be efficient on medium sized arrays as well. Finally we observe that experiments fit well the theory.

The number of comparisons performed by the pseudomedian-of-9 variant of quicksort proposed by Bentley and McIlroy [2] is analysed under the assump-

[1] Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

tion that all input elements are distinct. This quicksort algorithm, that is now the sorting method of FreeBSD, features interesting optimizations to reduce the number of assignments. Arrays filled with non-distinct values are also taken into account in the implementation. To assume that the keys to be sorted are all distinct simplifies the analysis. For the basic quicksort, the analysis with equal keys has been performed by Sedgewick [11].

1 The algorithm and its complexity recurrence equation

The sorting algorithm quicksort was invented in 1960 by C. Hoare [7]. Different versions of this algorithm can be found in [2,8,10]. The quicksort algorithm sorts the input array as follows. A pivot is chosen, the algorithm compares all the elements of the array (except the pivot) to the pivot. This divides the array into two subarrays such that the elements of the first one are smaller than the pivot and the elements of the second one are greater than the pivot. The two subarrays are then recursively sorted.

The choice of the pivot is essential, because the more central it is in the array, the more efficient is the splitting. But in order to keep the complexity reasonable, the decision has to be fast. The simplest idea is the standard quicksort, that takes an arbitrary element (e.g. the first) as the pivot. Another method is to choose a sample of three elements and take the median as the pivot. This is median-of-3 quicksort, that is studied in [8,12]. The last technique explained here is pseudomedian-of-9 quicksort, that proceeds as follows. We take three samples of three arbitrary elements (all nine elements are different), determine the median of each sample, and then choose the median of these three medians as the pivot. This technique is called pseudomedian, because the pivot is the fourth, fifth or sixth of the nine elements. The cost of determining the pseudomedian pivot is 32/3 comparisons, which correspond to determining the median of three elements ($\frac{2}{3} \times 3 + \frac{1}{3} \times 2$ comparisons) four times. This is much smaller than the cost of determining the exact median of 9 elements, which is 14.

In this paper, we assume that the array to be sorted is filled with distinct values, and that that the probability distribution is uniform on all the permutations of the values. The choice of the sample of elements is then indifferent. It is equivalent to consider that the array is filled with a random permutation, or that the elements in the sample are randomly chosen: because in both models, all the elements of the array have the same probability to be chosen in the sample.

The optimized algorithm implemented by Bentley and McIlroy is as follows: for an array smaller than 7, sort with the basic quicksort; for an array of size between 8 and 39 use median-of-3 quicksort; for larger arrays use pseudomedian-of-9 quicksort.

As complexity measure, we adopt the expected number of comparisons. For the analysis of the pseudomedian-of-9 quicksort algorithm we first have to determine the probability that the pivot is the kth element of the array, which is denoted by $\Pr(k)$, as the recurrence equation on the average complexity is

$$f_n = t_n + \sum_k \Pr(k)(f_{k-1} + f_{n-k}).$$

There t_n is the toll function, that describes the average cost of choosing the pivot $(32/3)$ and of comparing all the elements to the pivot $(n-1)$. The nine elements are represented by sorting the three samples by increasing order of their medians, then by sorting each sample. The letter k stands for the pivot. We obtain an array of the form:

$<k$	$<k$	$a?$	$<k$	k	$>k$	$b?$	$>k$	$>k$

For all values of a and b, the pivot chosen for this sample is k. We distinguish four different cases depending on the position of a and b with respect to k. If, for example $a < k$ and $b < k$ we are in the following situation:

$<k$	$<k$	$<k$	$<k$	k	$>k$	$<k$	$>k$	$>k$

There are $(k-1)(k-2)(k-3)\times(k-4)(n-k)3!\times(k-5)(n-k-1)(n-k-2)3\times 3!$ different choices of sample that lead to this configuration. We first have to choose 3 elements smaller than k, then one larger, one smaller and mix them with k, then one smaller and two larger, decide the position of the smaller, and finally mix the three samples. We remark that in this configuration the pivot is equal to k and is the sixth element of the sample, which is denoted $pivot = (k, 6^{th})$. The other configurations are treated the same way, and using the notation $(x)_k = x(x-1)\ldots(x-k-1)$, we obtain

$$\begin{aligned}
&\Pr(pivot = (k, 4^{th})) = 3(k-1)_3(n-k)_5(3!)^2/(n)_9;\\
&\Pr(pivot = (k, 5^{th})) = 10(k-1)_4(n-k)_4(3!)^2/(n)_9;\\
&\Pr(pivot = (k, 6^{th})) = 3(k-1)_5(n-k)_3(3!)^2/(n)_9.
\end{aligned}$$

By the symmetry properties of the sum, the recurrence equation becomes:

$$f_n = n + 29/3 + \frac{(3!)^2}{(n)_9} \sum_{k=4}^{n-5} 6(k-1)_3(n-k)_5 f_{k-1}$$
$$+ \frac{(3!)^2}{(n)_9} \sum_{k=5}^{n-4} 20(k-1)_4(n-k)_4 f_{k-1} + \frac{(3!)^2}{(n)_9} \sum_{k=6}^{n-3} 6(k-1)_5(n-k)_3 f_{k-1}.$$

This recurrence equation is used to obtain the differential equation stated in the following lemma.

Lemma 1 *The differential equation satisfied by the average-case complexity generating function $f(z) = \sum_n f_n z^n$ is*

$$(1-z)^9 f^{(9)}(z) = (1-z)^9 t(z) + 25920(1-z)^3 f^{(3)}(z) + 17280(1-z)^4 f^{(4)}(z) + 1296(1-z)^5 f^{(5)}(z),$$

where $t(z) = \frac{10!}{(1-z)^{11}} + \frac{26}{3} \frac{9!}{(1-z)^{10}}$, *and* $f^{(i)}$ *denotes* $\frac{\partial^i f(z)}{\partial z^i}$.

The proof is a direct application of standard rules for manipulating general functions [12] and is omitted. (The calculation can be checked automatically using the maple package GFUN [9].) This is an Euler equation, and the next section is devoted to the study of the solutions of such equations. We can already note that a particular solution of this differential equation is:

$$f(z) = \frac{\alpha}{(1-z)^2} \log_e \frac{1}{1-z} - \frac{26}{3}(1-z)^{-2},$$

with $\alpha = \frac{10!}{2311776} = 1.5697\ldots$ The first asymptotic term $1.569\, n \log n + o(n \log n)$ is produced by this particular solution. This is proved in section 3.

2 Euler Equations

An Euler equation is a linear differential equation $\mathcal{L}(f)$ of the form

$$a_d(1-z)^d f^{(d)}(z) + \cdots + a_1(1-z) f'(z) + a_0 f(z) = g(z)$$

where $a_i \in \mathbb{C}$; it is said to be homogeneous if $g(z) = 0$. This type of equation appears quite often in combinatorics, and especially here in the analysis of quicksort [3,4,8,12]. The Euler equations present the great advantage to admit homogeneous solutions that are explicit. We briefly give the general solution of an Euler equation in a particular case. The homogeneous part applied to the function $f(z) = (1-z)^{-\alpha}$ gives the relation $\mathcal{L}((1-z)^{-\alpha}) = I(\alpha)(1-z)^{-\alpha}$, where I is a polynomial of degree m called the indicial polynomial. This is easily verified with the formula for the derivatives of $(1-z)^{-\alpha}$:

$$\left((1-z)^{-\alpha}\right)^{(k)} = \frac{(\alpha+k-1)_k}{(1-z)^{\alpha+k}}.$$

Thus if α is a root of the indicial polynomial, then $(1-z)^{-\alpha}$ is a solution of the corresponding Euler equation. This polynomial has m roots in $\mathbb{C}$; if they are all distinct, we obtain m "free" solutions of the homogeneous equation, and as the equation is of degree m, these solutions form a basis of the space of solutions. (If the roots are not all distinct, a general solution is also easily found, but we do not need that case here.)

3 Asymptotic analysis

The complexity of the optimized quicksort (i.e., the Bentley-McIlroy's program) is governed by the behaviour of the pseudomedian-of-9 quicksort for arrays of size greater then forty. The recurrence equation of the optimized quicksort is the same as the recurrence equation of pseudomedian-of-9 quicksort, except for the first forty terms that serve as initial conditions. Managing forty initial conditions is not convenient, so we compute instead new "fake" initial conditions such that running the pseudomedian-of-9 quicksort recurrence on them yields the same sequence as the complexity of the optimized quicksort after for the first forty terms. These new initial conditions are computed easily by running the pseudomedian-of-9 quicksort recurrence backwards from values situated after the fortieth term: as the recurrence is of order 9, the terms from 41^{th} to 49^{th} can be used as initial conditions for this backward calculation. (We remark that these "fake" initial conditions involve very large rationals for the quicksort problem.)

We now study the asymptotic behaviour of the pseudomedian-of-9 quicksort recurrence, which is of degree 9, with 9 initial conditions. The differential equation coding this recurrence is stated in Lemma 1. The indicial polynomial of this Euler equation is

$$I(x) = (x-2)x(x+1)(x+2)(x^5+35x^4+515x^3+4165x^2+19188x+36576).$$

The roots of this polynomial are 2, 0, -1, -2, $-4.2\pm6.6i$, -4.5 and $-11\pm2.8i$ and are denoted by $\alpha_i, i \in 1,\dots,9$. As all the roots of this polynomial are distinct, the general homogeneous solution is $\sum_i \lambda_{\alpha_i}(1-z)^{-\alpha_i}$. As we already know a particular solution, a general solution of the inhomogeneous equation is:

$$f(z)=\frac{10!}{2311776}\frac{1}{(1-z)^2}\log_e\frac{1}{1-z}-\frac{26/3}{1-z}+\sum_i\frac{\lambda_{\alpha_i}}{(1-z)^{\alpha_i}}. \tag{1}$$

We so have nine unknown variables (λ_{α_i}, $1\le i\le 9$), and nine initial conditions (f_j, $0\le j\le 8$). By considering the coefficient of $z^j, j\in 0,\dots,8$ on both sides of the equation 1, we obtain 9 linear equations, where each f_j is expressed in terms of all the λ_{α_i}. Those 9 equations form a linear system can be written as a matrix equation:

$$F=A\Lambda+B. \tag{2}$$

where $F={}^t(f_0,\dots,f_8)$ (${}^tV$ is the transpose of V), $\Lambda={}^t(\lambda_{\alpha_1},\dots,\lambda_{\alpha_9})$ and the vector B contains the first terms of the expansion of the particular solution. The matrix A is defined by the series expansion of $\frac{1}{(1-z)^\alpha}$. This gives $A_{i,j}=\alpha_i(\alpha_i+1)\cdots(\alpha_i+j-1)\frac{1}{j!}$. The matrix A is very close to a Vandermonde matrix. We can decompose it as a product of a constant triangular matrix S of Stirling type by a Vandermonde matrix V. The Vandermonde matrix has an explicit inverse,

$$V^{-1}[i,j]=\frac{\sigma_{n-j-1}(i)(-1)^j}{\sum_{k=0}^{n-1}\sigma_k(i)(-1)^k\alpha_i^{n-k-1}}$$

where $\sigma_k(i)=\prod_{j=1}^k\alpha_{\epsilon_j}$ are the elementary symmetric functions with the additional condition $\epsilon_j\neq i$. This can be verified by a simple matrix multiplication. The matrix S has an inverse S^{-1} with coefficients equal to $(-1)^{i+j}\left\{{i\atop j}\right\}j!$, where the $\left\{{i\atop j}\right\}$ are the Stirling coefficients of second kind [1], according to the notation used in [6]. The inverse of the matrix A is then simply computed by $A^{-1}=V^{-1}S^{-1}$. The system (2) is solved by $\Lambda=A^{-1}(F-B)$.

The matrix A has the particularity that each line depends only on one root of the indicial polynomial. Thus each coefficient of the vector Λ depends only on one root of the polynomial I. More precisely, all the coefficients of Λ are the same rational function, but evaluated at a different root of I. So Λ is ${}^t(\lambda(\alpha_1),\dots,\lambda(\alpha_9))$ where the function $\lambda(\alpha)$ is a rational function of small degree and very large coefficients. A consequence is that the generating function f is determined as

$$\sum_{I(\alpha)=0}\frac{\lambda(\alpha)}{(1-z)^\alpha}+\frac{12600}{8027}\frac{1}{(1-z)^2}\log_e\frac{1}{1-z}-\frac{26/3}{1-z}.$$

For a power series $f(z)=\sum_n f_nz^n$, the notation $[z^n]f(z)$ stands for f_n, the coefficient of z^n in f. To get the asymptotic expansion of $f(z)$ we use the dictionary [5] stating that:

$$[z^n](1-z)^{-\alpha}=\frac{n^{\alpha-1}}{\Gamma(\alpha)}(1+O(1/n)),$$

$$[z^n](1-z)^{-2}\log\frac{1}{1-z} = n\log n + (\gamma-1)n + \log n + \frac{1}{2} + \gamma + O(1/n).$$

We thus have the theorem:

Theorem 2 *The asymptotic expansion of f_n is*

$$f_n = \frac{12600}{8027} n\log_e n + \left((\gamma-1)\frac{12600}{8027} - \lambda(2)\right) n + \\ \frac{12600}{8027}\log_e n + \left(\left(\frac{1}{2}+\gamma\right)\frac{12600}{8027} - \frac{26}{3} - \lambda(2)\right) + O\left(\frac{1}{n}\right).$$

As $\lambda(2) = 0.3727\ldots$ and γ is the Euler constant, the expansion with approximate numerical constants is

$$f_n = 1.5697\, n\log_e n - 1.0363\, n + 1.5697\, \log_e n - 7.3484 + O(1/n).$$

This is close to what Bentley and McIlroy found empirically, namely $f_n = 1.5783\, n\log_e n - 0.74\, n$ obtained by a weighted least-squares regression.

4 Ideal quicksort

To create an "ideal" quicksort out of the three basic quicksorts used by Bentley and McIlroy, we have to decide the cut-off values that determine the switch between the three pivot strategies. Under our criterion, this only depends on the recurrences satisfied by the complexities defined as the average number of comparisons. For the basic quicksort it is $f_n = n - 1 + \frac{2}{n}\sum_{k=0}^{n-1} f_k$ and for median-of-3 quicksort, $f_n = n+5/3+\sum_{k=2}^{n-1} 6(n-k)(k-1)\frac{f_{k-1}+f_{n-k}}{n(n-1)(n-2)}$, see [8,12].

First, in order to determine the cut-off value between quicksort and median-of-3 quicksort, we first observe that it has to be larger than 3, and then find an upper bound, for example 20 because then median-of-3 quicksort is always better. By comparing all the algorithms created with a cut-off value between 3 and 20 exhaustively, we observe that one of the variants is better than all the others. This determines the cut-off value, here 15. We then proceed in the same way to get the cut-off value between the median-of-3 quicksort obtained previously and pseudomedian-of-9 quicksort. The cut-off values that we obtain are

$n < 15$	standard quicksort
$15 \le n < 86$	median-of-3 quicksort
$86 \le n$	pseudomedian-of-9 quicksort

These cut off values (15,86) are slightly different from the cut-off values of Bentley and McIlroy (7,39). Using these cut-off values, we obtain an optimal sorting algorithm, with respect to the number of comparisons, among all the sorting algorithms based on quicksort, median-of-3 quicksort, and pseudomedian-of-9 quicksort. Its complexity in terms of comparisons is

$$f_n = 1.5697\, n \log_e n - 1.1512\, n + 1.5697\, \log_e n - 7.4633 + o(1),$$

which is slightly better than Bentley and McIlroy's algorithm. For an array of size 1000, this optimal quicksort above is 1.2% better than Bentley and McIlroy's, and 12% better than the basic quicksort. However we have to keep in mind that other parameters are important in the running time of the algorithm like the number of exchanges, that we did not consider. We also remark that these complexity functions are not far from the theoretical bound of $\log_e n! \sim 1.4426\, n \log_e n - 1.44\, n$.

There are many other pivot strategies. For example choose a sample of size 3^k, select 3^{k-1} medians, 3^{k-2} medians of medians and so on, until only one, the pivot is left; or choose m pivots, $m > 1$, and then divide the array into $m+1$ subarrays; or select m pivots in a sample of size greater than m. The analysis, either on average or distribution, of these variants involves very often Euler equations. An informative discussion of Euler equations in the context of analysis of algorithms is provided by recent and independent work of Chern and Hwang [3] and Fill [4].

References

[1] Abramowitz, M., and Stegun, I. A. *Handbook of Mathematical Functions.* Dover, 1973. A reprint of the tenth National Bureau of Standards edition, 1964.

[2] Bentley, J. L., and McIlroy, D. Engineering a sort function. *Software—Practice and Experience 23*, 11 (1993), 1249–1265.

[3] Chern, H.-H., and Hwang, H.-K. Phase changes in random m-ary search trees and generalized quicksort. Preprint, 2001.

[4] Fill, J. A. A unified (general) treatment of (a certain class of additive-type) tree functionals under the random permutation model. Preprint, 2001.

[5] Flajolet, P., and Odlyzko, A. M. Singularity analysis of generating functions. *SIAM Journal on Algebraic and Discrete Methods 3*, 2 (1990), 216–240.

[6] Graham, R., Knuth, D., and Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science*, second ed. Addison Wesley, 1994.

[7] Hoare, C. A. R. Quicksort. *Computer J. 5*, 1 (1962), 10–15.

[8] KNUTH, D. E. *The Art of Computer Programming*, 2nd ed., vol. 3: Sorting and Searching. Addison-Wesley, 1998.

[9] SALVY, B., AND ZIMMERMANN, P. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software 20*, 2 (1994), 163–177.

[10] SEDGEWICK, R. *Algorithms*, second ed. Addison–Wesley, 1988.

[11] SEDGEWICK, R. Quicksort with equal keys. *SIAM Journal on Computing 6*, 2 (June 1977), 240–267.

[12] SEDGEWICK, R., AND FLAJOLET, P. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Publishing Company, 1996.