11/2 (2013), 195–210
tmcs@math.klte.hu
http://tmcs.math.klte.hu

Teaching Mathematics and Computer Science

A Didactic Analysis of Merge Sort

Christian Rinderknecht

Abstract. Due to technical difficulties, educators teaching merge sort often avoid the analysis of the cost in the general and average cases. Using basic discrete mathematics, elementary real analysis and mathematical induction, we propose a self-contained derivation of bounds $\alpha n \log_2 n + \beta n + \gamma$ in all cases. Independent of any programming language or pseudo-code, supported by intuitive figures, it is suitable for informatics students interested in the analysis of algorithms. It is also a good exercise in showing that induction allows us to actually discover constants, instead of simply checking them a posteriori.

Key words and phrases: didactics of informatics, analysis of algorithms, cost analysis, merging; merge sort, enumerative combinatorics.

ZDM Subject Classification: M55, N75, K25.

Knuth [1] reports that the first computer program, designed in 1945 by the mathematician John von Neumann, was a sorting algorithm, nowadays called *merge sort.* It is amongst the most widely taught sorting algorithms because it epitomises the important solving strategy known as *divide and conquer*: the input is split, each non-trivial part is recursively processed and the partial solutions are finally combined to form the complete solution. Whilst merge sort is not difficult to program, determining its efficiency, by means of a *cost* function, requires

The research was carried out as part of the EITKIC_12-1-2012-0001 project, which is supported by the Hungarian Government, managed by the National Development Agency, financed by the Research and Technology Innovation Fund and is performed in cooperation with the EIT ICT Labs Budapest Associate Partner Group (`ictlabs.elte.hu`).

advanced mathematical knowledge. Therefore, most textbooks [2, 3] only show how to find the order of growth of an upper bound of the cost from recurrences it satisfies when n is a power of 2, but the general case is often not presented in the main chapters, or not at all, and precise analytic solutions are extremely difficult, making use of complex analysis [4, 5, 6]. Amongst the several variants of merge sort [7, 8], we will be dealing here with the most popular, called *top-down merge sort*.

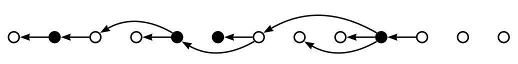
We derive lower bounds $\alpha n \log_2 n + \beta n + \gamma$ and upper bounds $\alpha n \lg \log_2 n + \beta' n + \gamma'$ for the extremum and mean costs, relying only on basic discrete mathematics, intuitive figures, elementary real analysis and mathematical induction. This is much more precise than using Bachmann's notation and state $O(n \log_2 n)$ in all cases. Worse, this notation is often misconstrued by students, and occasionally by professionals, when it is used on the main term of the asymptotic expansion, as noted by Knuth already in 1976: "Unfortunately, people have occasionally been using the O-notation for lower bounds, for example when they reject a particular sorting method 'because its running time is $O(n^2)$.' I have seen instances of this in print quite often [...]" [9].

1. Merging

Merging consists in combining two ordered series of keys into one ordered series. Without loss of generality, we shall be only interested in sorting keys in increasing order, like merging $(10, 12, 17)$ and $(13, 14, 16)$ results in (10,12,13,14,16, 17) One way to achieve this consists in comparing the two smallest keys, output the smallest and repeat the procedure until one of the series becomes empty, in which case the other is wholly appended. We have (compared keys underlined):

$$\begin{cases} \underline{10}\ 12\ 17 \\ \underline{13}\ 14\ 16 \end{cases} \rightarrow 10 \begin{cases} \underline{12}\ 17 \\ \underline{13}\ 14\ 16 \end{cases} \rightarrow 10\ 12 \begin{cases} \underline{17} \\ \underline{13}\ 14\ 16 \end{cases} \rightarrow 10\ 12\ 13 \begin{cases} \underline{17} \\ \underline{14}\ 16 \end{cases} \text{etc.}$$

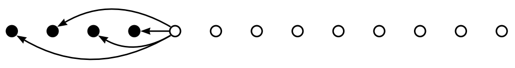
We depict a key from one series as a *white node* ($\circ$) and a key from the other as a *black node* ($\bullet$). Nodes of these kinds are printed in a horizontal line, the leftmost being the smallest. Comparisons are always performed between black and white nodes and are represented as *edges*, for instance:

 (1)

An incoming arrow means that the node is smaller than the other end of the edge, so all edges point leftward and the number of comparisons is the number of nodes with an incoming edge. This number is none other than the *cost* of merging.

Minimum cost

In (1), there are two consecutive white nodes without any edges at the right end, which suggests that the more keys from one series we have at the end of the result, the fewer comparisons we needed for merging: the minimum cost is achieved when *the shorter series comes first in the result.* Here, the cost is the number of black nodes:

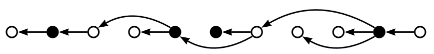


The minimum cost $\mathcal{B}^{\bowtie}_{m,n}$ when merging series of length m and n is

$$\mathcal{B}^{\bowtie}_{m,n} = \min\{m, n\}. \tag{2}$$

Maximum cost

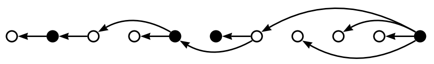
We can increase the number of comparisons with respect to $m+n$ by removing in (1) those rightmost nodes in the result *that are not compared*:

 (3)

This maximises comparisons because all nodes, but the last, are the destination of an edge. The maximum cost $\mathcal{W}^{\bowtie}_{m,n}$ is the maximum number of comparisons when merging series of length m and n:

$$\mathcal{W}^{\bowtie}_{m,n} = m + n - 1. \tag{4}$$

Interchanging the two rightmost nodes in (3) leaves $m + n - 1$ invariant, so the maximum number of comparisons occurs when *the last two keys of the result come from two series:*



Average cost

The average of the costs of merging all pairs of series of given lengths defines the average cost, assuming that keys are not repeated. Consider *Figure* 1, with 35 comparisons and 10 results, so the average cost is $35/10 = 7/2$. In general, there are $\binom{m+n}{n}$ ways to interleave m white nodes with n black nodes, as it is the

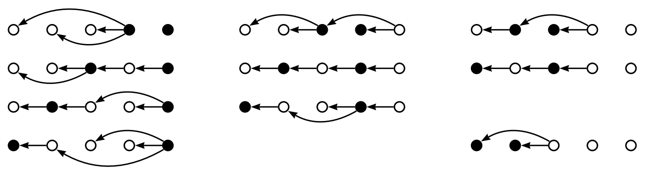


Figure 1. All mergers with $m = 3$ ($\circ$) and $n = 2$ ($\bullet$)

same as the number of ways to pick n nodes amongst $m + n$. The total number $K_{m,n}$ of comparisons needed to merge m and n keys in all possible manners with our algorithm is the number of nodes with incoming edges. Let $\overline{K}_{m,n}$ be the total number of nodes *without* incoming edges, circled in *Figure* 2.

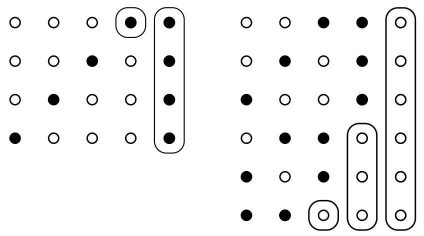


Figure 2. Counting vertically

This figure has been obtained by moving the third column of *Figure* 1 below the second column and by removing the edges. Since, for each merger, there are $m+n$ nodes and each has an incoming edge or not, and because there are $\binom{m+n}{n}$ mergers, we have $K_{m,n} + \overline{K}_{m,n} = (m + n)\binom{m+n}{n}$. It is simple to characterise the circled nodes: they make up the longest, rightmost contiguous series of nodes of the same colour. Since there are only two colours, the problem of determining the total number $W_{m,n}$ of white circled nodes is symmetric to the determination of the total number $B_{m,n}$ of black circled nodes, that is, $B_{m,n} = W_{n,m}$. We also obviously have $\overline{K}_{m,n} = W_{m,n} + B_{m,n} = W_{m,n} + W_{n,m}$, therefore

$$K_{m,n} = (m + n)\binom{m + n}{n} - W_{m,n} - W_{n,m}.$$

We can decompose $W_{m,n}$ by counting the circled white nodes *vertically.* In *Figure* 2, $W_{3,2}$ is the sum of the numbers of mergers with one, two and three ending circled white nodes: $W_{3,2} = 6 + 3 + 1 = 10$. The first column yields $B_{3,2} = 4 + 1 = 5$. In general, the number of mergers with one ending circled white node is the number of ways to combine n black nodes with $m-1$ white nodes: $\binom{n+m-1}{n}$. Similarly, the number of mergers with two ending circled white nodes is $\binom{n+m-2}{n}$, etc. Therefore, using standard binomial identities [2], we derive

$$W_{m,n} = \sum_{j=0}^{m-1} \binom{n+j}{n} = \sum_{j=0}^{m-1} \binom{n+j}{j} = \binom{n+m}{m-1} = \binom{m+n}{n+1}.$$

Then $K_{m,n} = (m+n)\binom{m+n}{n} - \binom{m+n}{n+1} - \binom{m+n}{m+1}$. By definition, the average cost $\mathcal{A}^{\bowtie}_{m,n}$ is the ratio of $K_{m,n}$ by $\binom{m+n}{n}$, therefore

$$\mathcal{A}^{\bowtie}_{m,n} = m + n - \frac{m}{n+1} - \frac{n}{m+1}. \tag{5}$$

2. Sorting 2^n keys

Merging can be used to sort *one* series of keys as follows. The initial series of keys is split in two, then the two pieces are split again etc. until singletons remain. These are then merged pairwise etc. until only one series remains, which is inductively sorted, since a singleton is a sorted series on its own and the merger of two series is sorted if they are both sorted. The previous scheme leaves open the choice of a splitting strategy and, perhaps, the most intuitive way is to cut in two halves, which works well in the case of 2^p keys. We will see in the next section how to deal with the general case.

Here, let us consider in *Figure* 3 all the mergers and their relative order to sort the series $(7,3,5,1,6,8,4,2)$. We name this structure a *merge tree*, because the nodes of the tree are sorted series, either singletons or resulting from the merging of its two children. The root logically holds the result. The merge tree is best understood from a bottom-up, level by level examination. Let us note

$$\begin{tikzcd}
 & & & (1,2,3,4,5,6,7,8) & & & & \\
 & (1,3,5,7) \arrow[urr] & & & & (2,4,6,8) \arrow[ull] & & \\
(3,7) \arrow[ur] & & (1,5) \arrow[ul] & & (6,8) \arrow[ur] & & (2,4) \arrow[ul] & \\
(7) \arrow[u] & (3) \arrow[ul] & (5) \arrow[u] & (1) \arrow[ul] & (6) \arrow[u] & (8) \arrow[ul] & (4) \arrow[u] & (2) \arrow[ul]
\end{tikzcd}$$

Figure 3. Sorting $(7,3,5,1,6,8,4,2)$

$\mathcal{C}_{2^p}$ the number of comparisons to sort 2^p keys and consider a merge tree with 2^{p+1} leaves. It is made of two immediate subtrees with 2^p leaves and the root holds 2^{p+1} keys. Therefore

$$\mathcal{C}_1 = 0, \quad \mathcal{C}_{2^{p+1}} = 2 \cdot \mathcal{C}_{2^p} + \mathcal{C}^{\bowtie}_{2^p,2^p}.$$

Unrolling the recursion, we arrive at

$$\mathcal{C}_{2^{p+1}} = 2^p \sum_{k=0}^{p} \frac{1}{2^k} \mathcal{C}^{\bowtie}_{2^k,2^k}. \tag{6}$$

Minimum cost

When the given series is already sorted, either in increasing or decreasing order, the number of comparisons is minimum. In fact, given a minimum-comparison merge tree, any exchange of two subtrees whose roots are merged leaves the number of comparisons invariant. This happens because the merge tree is built bottom-up and the number of comparisons is a symmetric function. Let us note $\mathcal{B}_{2^p}$ the minimum number of comparisons to sort 2^p keys. From equations (2) and (6), we draw

$$\mathcal{B}_{2^p} = 2^{p-1} \sum_{k=0}^{p-1} \frac{1}{2^k} \mathcal{B}^{\bowtie}_{2^k,2^k} = p2^{p-1}. \tag{7}$$

Maximum cost

Just as with the best case, constructing a maximum-comparison merge sort is achieved by making worst cases for all the subtrees, like $(7,3,5,1,4,8,6,2)$. Let $\mathcal{W}_{2^p}$ be the maximum number of comparisons for sorting 2^p keys. From equations (4) and (6), we deduce

$$\mathcal{W}_{2^p} = 2^{p-1} \sum_{k=0}^{p-1} \frac{1}{2^k} \mathcal{W}^{\bowtie}_{2^k,2^k} = (p-1)2^p + 1. \tag{8}$$

Average cost

For a given series, all permutations of which are equally likely, the average cost of sorting it by merging is obtained by considering the average costs of all the subtrees of the merge tree: all the permutations of the keys are considered for a given length. Therefore, equation (6) is satisfied by the average cost $\mathcal{A}_{2^p}$, that is, the average number of comparisons for sorting 2^p keys. Besides, equation (5) yields

$$\mathcal{A}^{\bowtie}_{n,n} = 2n - 2 + \frac{2}{n+1}.$$

Together with equation (6), we further draw, for $p > 0$,

$$\begin{aligned}
\mathcal{A}_{2^p} &= 2^{p-1}\sum_{k=0}^{p-1}\frac{1}{2^k}\mathcal{A}^{\bowtie}_{2^k,2^k} = 2^p\sum_{k=0}^{p-1}\frac{1}{2^k}\left(2^k - 1 + \frac{1}{2^k+1}\right)\\
&= 2^p\left(p - \sum_{k=0}^{p-1}\frac{1}{2^k} + \sum_{k=0}^{p-1}\frac{1}{2^k(2^k+1)}\right),\\
\mathcal{A}_{2^p} &= 2^p\left(p - \sum_{k=0}^{p-1}\frac{1}{2^k} + \sum_{k=0}^{p-1}\left(\frac{1}{2^k} - \frac{1}{2^k+1}\right)\right) = p2^p - 2^p\sum_{k=0}^{p-1}\frac{1}{2^k+1}\\
&= p2^p - 2^p\sum_{k\geqslant 0}\frac{1}{2^k+1} + 2^p\sum_{k\geqslant p}\frac{1}{2^k+1} = p2^p - \alpha 2^p + \sum_{k\geqslant 0}\frac{1}{2^k+2^{-p}},
\end{aligned}$$

where $\alpha := \sum_{k\geqslant 0}\frac{1}{2^k+1} \simeq 1.264500$ is irrational [10]. Since $0 < 2^{-p} < 1$, we have $1/(2^k+1) < 1/(2^k+2^{-p}) < 1/2^k$ and we conclude

$$(p-\alpha)2^p + \alpha < \mathcal{A}_{2^p} < (p-\alpha)2^p + 2. \tag{9}$$

The uniform convergence of the series $\sum_{k\geqslant 0}\frac{1}{2^k+2^{-p}}$ allows us to interchange the limits on k and p and deduce that $\mathcal{A}_{2^p} - (p-\alpha)2^p - 2 \to 0^-$, as $p \to \infty$. In other words, $\mathcal{A}_{2^p}$ is best approximated by its upper bound, for large values of p.

3. Merge Sort

In general, merge sort consists in splitting a series in two series of *equal or almost equal* lengths, which are in turn recursively sorted, except for the singleton, and merged. The number of comparisons $\mathcal{C}_n$ hence satisfies

$$\mathcal{C}_0 = \mathcal{C}_1 = 0, \quad \mathcal{C}_n = \mathcal{C}_{\lfloor n/2\rfloor} + \mathcal{C}_{\lceil n/2\rceil} + \mathcal{C}^{\bowtie}_{\lfloor n/2\rfloor,\lceil n/2\rceil}, \tag{10}$$

where $\lfloor x\rfloor$ (*floor of* x) is the greatest integer less than or equal to x, and $\lceil x\rceil$ (*ceiling of* x) is the least integer greater than or equal to x. So we have $n = \lfloor n/2\rfloor + \lceil n/2\rceil$.

Minimum cost

We have $\mathcal{B}_n = \mathcal{B}_{\lfloor n/2\rfloor} + \mathcal{B}_{\lceil n/2\rceil} + \lfloor n/2\rfloor$ from equations (2) and (10). Let $\Delta_n := \mathcal{B}_{n+1} - \mathcal{B}_n$ and find some constraints on it. Because of the floor and ceiling functions of $n/2$, we consider two complementary cases.

- If $n = 2p$, then $\mathcal{B}_{2p} = 2 \cdot \mathcal{B}_p + p$ and $\mathcal{B}_{2p+1} = \mathcal{B}_p + \mathcal{B}_{p+1} + p$. Therefore, $\Delta_{2p} = \Delta_p$. Also $\Delta_0 = 0$.
- If $n = 2p + 1$, then $\mathcal{B}_{2p+2} = 2 \cdot \mathcal{B}_{p+1} + p + 1$. Thus, $\Delta_{2p+1} = \Delta_p + 1$.

2^0	1
2^1	10
	11
2^2	100
	101
	110
	111
2^3	1000
	1001
	1010
	1011
	1100
	1101
	1110
	1111
$\vdots$	$\vdots$
$2^{\lfloor \lg n \rfloor}$	$\dots$
	$\vdots$
	n

Figure 4

If we think in terms of binary representations, Δ_n is the number of 1-bits or, equivalently, the sum of the bits of n. It is usually noted ν_n and called *bit sum*. So $\mathcal{B}_{n+1} = \mathcal{B}_n + \nu_n$, therefore

$$\mathcal{B}_n = \sum_{k=0}^{n-1} \nu_k. \tag{11}$$

Equation (7) is $\mathcal{B}_{2^p} = \frac{1}{2}p2^p$, that is, $\mathcal{B}_n = \frac{1}{2}n \log_2 n$ when $n = 2^p$. This should prompt us to look, like McIlroy [11], for an additional linear term in the general case, that is, the greatest real constants a and b such that, for $n \geqslant 2$,

$$\mathsf{L}(n)\colon \tfrac{1}{2}n \log_2 n + an + b \leqslant \mathcal{B}_n, \tag{12}$$

where $\log_2 n$ is the binary logarithm of n. The base case is simply $\mathsf{L}(2)\colon 2a + b \leqslant 0$. The most obvious way to structure the inductive argument is to follow the definition of $\mathcal{B}_n$ when $n = 2p$ and $n = 2p + 1$, but a bound on $\mathcal{B}_{2p+1}$ would rely on bounds on $\mathcal{B}_p$ and $\mathcal{B}_{p+1}$, compounding imprecision. Instead, if we could have at least one exact value from which to inductively build the bound, we would gain accuracy. Therefore, we may expect a better bound if we can find a decomposition of $\mathcal{B}_{2^p+i}$, where $0 < i \leqslant 2^p$, in terms of $\mathcal{B}_{2^p}$ (exact) and $\mathcal{B}_i$. This is easy if we count the bits in *Figure* 5, which is the same as the table in *Figure* 4, where $n = 2^p + i$. (Keep in mind that $\mathcal{B}_n$ is the sum of the bits up to $n - 1$, as seen in equation (11).) We find:

$$\mathcal{B}_{2^p+i} = \mathcal{B}_{2^p} + \mathcal{B}_i + i.$$

(The term i is the sum of the leftmost bits.) Therefore, let us assume $\mathsf{L}(n)$, for all $1 \leqslant n \leqslant 2^p$, and prove $\mathsf{L}(2^p + i)$, for all $0 < i \leqslant 2^p$. The induction principle entails then that $\mathsf{L}(n)$ holds for all $n \geqslant 2$. The inductive step $\mathsf{L}(2^p + i)$ should give us the opportunity to maximise the constants a and b. Let $m = 2^p$. Using $\mathcal{B}_{2^p} = \frac{1}{2}p2^p$ from equation (7) and the inductive hypothesis $\mathsf{L}(i)$, we have

$$\tfrac{1}{2}m \log_2 m + (\tfrac{1}{2}i \log_2 i + ai + b) + i \leqslant \mathcal{B}_m + \mathcal{B}_i + i = \mathcal{B}_{m+i}. \tag{13}$$

Bits of n		n
0	0 $\ldots$ 0	0
$\vdots$	$\mathcal{B}_{2^p}$	$\vdots$
0	1 $\ldots$ 1	$2^p - 1$
1	0 $\ldots$ 0	2^p
$\vdots$	$\mathcal{B}_i$	$\vdots$
1	$\ldots$	$2^p + i - 1$
		$2^p + i$

Figure 5. $\mathcal{B}_{2^p+i} = \mathcal{B}_{2^p} + \mathcal{B}_i + i$

We need now to find a and b such that the inductive step $L(m+i)$ holds as well, that is, $\frac{1}{2}(m+i)\log_2(m+i) + a(m+i) + b \leqslant \mathcal{B}_{m+i}$. Using (13), this is implied by

$$\tfrac{1}{2}(m+i)\log_2(m+i) + a(m+i) + b \leqslant \tfrac{1}{2}m\log_2 m + (\tfrac{1}{2}i\log_2 i + ai + b) + i.$$

We can already notice that this inequality is equivalent to

$$\tfrac{1}{2}m\log_2(m+i) + \tfrac{1}{2}i\log_2(m+i) + am \leqslant \tfrac{1}{2}m\log_2 m + \tfrac{1}{2}i\log_2 i + i. \tag{14}$$

But $\frac{1}{2}m\log_2(m+i) > \frac{1}{2}m\log_2 m$ and $\frac{1}{2}i\log_2(m+i) > \frac{1}{2}i\log_2 i$, therefore the constant a we are seeking must satisfy $am \leqslant i$ for all $0 < i < m$, hence $a < 0$.

We extend i over the real numbers by defining $i = x2^p = xm$, where x is a real number such that $0 < x \leqslant 1$. By replacing i by xm in inequality (14), we obtain

$$\tfrac{1}{2}(1+x)\log_2(1+x) + a \leqslant \tfrac{1}{2}x\log_2 x + x.$$

Let $\Phi(x) := \frac{1}{2}x\log_2 x - \frac{1}{2}(1+x)\log_2(1+x) + x$. Then, this is equivalent to $a \leqslant \Phi(x)$.

The function Φ can be continuously extended at 0, as $\lim_{x\to 0} x\log_2 x = 0$, and it is differentiable on the interval $]0,1]$:

$$\frac{d\Phi}{dx} = \frac{1}{2}\log_2\frac{4x}{x+1}. \tag{15}$$

The root of $d\Phi/dx = 0$ is $1/3$, and the derivative is negative before, and positive after. Therefore, $a_{\max} := \min_{0<x\leqslant 1}\Phi(x) = \Phi(\frac{1}{3}) = -\frac{1}{2}\log_2\frac{4}{3}$. The base case was $b \leqslant -2a$, therefore $b_{\max} := -2a_{\max} = \log_2\frac{4}{3}$. Finally,

$$\tfrac{1}{2}n\log_2 n - \left(\tfrac{1}{2}\log_2\tfrac{4}{3}\right)n + \log_2\tfrac{4}{3} \leqslant \mathcal{B}_n, \tag{16}$$

where $\frac{1}{2}\log_2\frac{4}{3} \simeq 0.2075$ and $\log_2\frac{4}{3} \simeq 0.415$. Importantly, the lower bound is tight if $x = 1/3$, that is, when $2^p + i = 2^p + x2^p = (1 + 1/3)2^p = 2^{p+2}/3$, or, in general, $2^k/3$. The nearest integers are $\lfloor 2^k/3 \rfloor$ and $\lceil 2^k/3 \rceil$, so we should find out which one minimises $\mathcal{B}_n - \frac{1}{2}n\log_2(\frac{3}{4}n)$, because we have $\frac{1}{2}n\log_2 n - \left(\frac{1}{2}\log_2\frac{4}{3}\right)n = \frac{1}{2}n\log_2(\frac{3}{4}n)$. It is tedious to prove that the lower bound is tight if $n = 2$ (from the base case) and is otherwise the sharpest when $n = (1010\ldots01)_2$ or $n = (1010\ldots1011)_2$. As a whole, these values constitute the *Jacobsthal sequence*, defined as

$$J_0 = 0;\ J_1 = 1;\ J_{n+2} = J_{n+1} + 2J_n,\ \text{for } n \geqslant 0.$$

Let us use now the same inductive approach to find a good upper bound to $\mathcal{B}_n$, that is, we want to minimise the real constants a' and b' such that, for $n \geqslant 2$,

$$\mathcal{B}_n \leqslant \tfrac{1}{2}n\log_2 n + a'n + b'.$$

The only difference with the search for the lower bound is that inequalities are reversed, so we want

$$\Phi(x) \leqslant a',\ \text{where } \Phi(x) := \tfrac{1}{2}x\log_2 x - \tfrac{1}{2}(1+x)\log_2(1+x) + x.$$

Here, we need to find the maximum of Φ on the closed interval $[0,1]$. The two positive roots of Φ are 0 and 1, and Φ is negative between them (see equation (15)). Therefore $a'_{\min} := \max_{0<x\leqslant 1}\Phi(x) = \Phi(1) = 0$. From the base case, we deduce $b'_{\min} = -2a_{\min} = 0$. Therefore, we have the bounds

$$\tfrac{1}{2}n\log_2 n - \left(\tfrac{1}{2}\log_2\tfrac{4}{3}\right)n + \log_2\tfrac{4}{3} \leqslant \mathcal{B}_n \leqslant \tfrac{1}{2}n\log_2 n. \tag{17}$$

The upper bound is clearly tight when $n = 2^p$ because of equation (7). Obviously, $\mathcal{B}_n \sim \frac{1}{2}n\log_2 n$, where $f(x) \sim g(x)$ means $\lim_{x\to\infty} f(x)/g(x) = 1$, but if we were only interested in this asymptotic result, Bush [12] gave a simpler counting argument on the bits in *Figure* 4. Delange [13] investigated $\mathcal{B}_n$ by means of advanced real analysis and showed that $\mathcal{B}_n = \frac{1}{2}n\log_2 n + F_0(\log_2 n)\cdot n$, where F_0 is a continuous, nowhere differentiable function of period 1, and whose Fourier series shows the mean value to be about -0.145599.

Maximum cost

The maximum number of comparisons satisfies

$$\mathcal{W}_0 = \mathcal{W}_1 = 0, \qquad \mathcal{W}_n = \mathcal{W}_{\lfloor n/2\rfloor} + \mathcal{W}_{\lceil n/2\rceil} + \mathcal{W}^{\bowtie}_{\lfloor n/2\rfloor,\lceil n/2\rceil}.$$

Equations (4) and (10) yield $\mathcal{W}_n = \mathcal{W}_{\lfloor n/2\rfloor} + \mathcal{W}_{\lceil n/2\rceil} + n - 1$ and

$$\mathcal{W}_0 = \mathcal{W}_1 = 0;\quad \mathcal{W}_{2p} = 2\mathcal{W}_p + 2p - 1,\quad \mathcal{W}_{2p+1} = \mathcal{W}_p + \mathcal{W}_{p+1} + 2p.$$

Let the difference of two successive terms be $\Delta_n := \mathcal{W}_{n+1} - \mathcal{W}_n$. If we know Δ_n, we know $\mathcal{W}_n$ because $\sum_{k=1}^{n-1} \Delta_k = \sum_{k=1}^{n-1} \mathcal{W}_{k+1} - \sum_{k=1}^{n-1} \mathcal{W}_k = \mathcal{W}_n - \mathcal{W}_1 = \mathcal{W}_n$. We remark that

- if $n = 2p$, then $\Delta_{2p} = \Delta_p + 1$,
- otherwise $n = 2p + 1$ and $\mathcal{W}_{2p+2} = 2 \cdot \mathcal{W}_{p+1} + 2p + 1$, so $\Delta_{2p+1} = \Delta_p + 1$.

In summary, $\Delta_0 = 0$ and $\Delta_n = \Delta_{\lfloor n/2 \rfloor} + 1$. If we start unravelling the recurrence, we get $\Delta_n = \Delta_{\lfloor \lfloor n/2 \rfloor /2 \rfloor} + 2 = \Delta_{\lfloor n/2^2 \rfloor} + 2$. (See [14, Exercise 1.2.4.35].) By iteration, we deduce $\Delta_n = m$, with m being the largest natural number such that $\lfloor n/2^m \rfloor = 0$. In other words, m is the number of bits in the binary notation of n. That number is found by setting $n := \sum_{i=0}^{m-1} b_i 2^i$, where the $b_i \in \{0, 1\}$ are the bits and $b_{m-1} = 1$. Then

$$2^{m-1} \leqslant n < 2^m \Rightarrow m - 1 \leqslant \log_2 n < m \Rightarrow m = \lfloor \log_2 n \rfloor + 1 = \Delta_n. \tag{18}$$

We already know that $\mathcal{W}_n = \sum_{k=1}^{n-1} \Delta_k$, therefore, with (18), we conclude that

$$\mathcal{W}_n = \sum_{k=1}^{n-1} (\lfloor \log_2 k \rfloor + 1). \tag{19}$$

Whilst the minimum cost is the number of 1-bits up to $n-1$, we find now that the maximum cost is the total number of bits up to $n-1$. Informally, this leads us to bet that $\mathcal{W}_n \sim 2 \cdot \mathcal{B}_n \sim n \log_2 n$, since we would expect the number of 0-bits and 1-bits to be the same in average. Consider again the bit table in *Figure* 4. The greatest power of 2 smaller than n is $2^{\lfloor \log_2 n \rfloor}$ because it is the binary number $(10\ldots0)_2$ having the same number of bits as n; it thus appears in the same section of the table as n. The trick consists in counting the bits in *columns*, from top to bottom, and leftwards. In the rightmost column, we find n bits. In the second column, from the right, we find $n - 2^1 + 1$ bits. The third from the right contains $n - 2^2 + 1$ bits etc. until the leftmost column containing $n - 2^{\lfloor \log_2 n \rfloor} + 1$ bits. The total number of bits in the table is

$$\sum_{k=1}^{n} (\lfloor \log_2 k \rfloor + 1) = \sum_{k=0}^{\lfloor \log_2 n \rfloor} (n - 2^k + 1) = (n+1)(\lfloor \log_2 n \rfloor + 1) - 2^{\lfloor \log_2 n \rfloor + 1} + 1.$$

Let $n := (b_{m-1} \ldots b_0)_2$, then $2^{m-1} \leqslant n \leqslant 2^m - 1$ and $2^{m-1} < 2^{m-1} + 1 \leqslant n + 1 \leqslant 2^m$, so $m - 1 < \log_2(n+1) \leqslant m$, that is, $m = \lceil \log_2(n+1) \rceil$. Using eq. (18), we deduce $1 + \lfloor \log_2 n \rfloor = \lceil \log_2(n+1) \rceil$. As a consequence, equation (19) can be rewritten as

$$\mathcal{W}_0 = \mathcal{W}_1 = 0, \qquad \mathcal{W}_n = n \lceil \log_2 n \rceil - 2^{\lceil \log_2 n \rceil} + 1. \tag{20}$$

This equation is subtler than it seems, due to the periodicity hidden in $2^{\lceil \log_2 n \rceil}$. Depending on whether $n = 2^p$ or not, two cases arise:

- if $n = 2^p$, then $\mathcal{W}_n = n \log_2 n - n + 1$;
- otherwise, we have $\lceil \log_2 n \rceil = \lfloor \log_2 n \rfloor + 1 = \log_2 n - \{\log_2 n\} + 1$ and $\mathcal{W}_n = n \log_2 n + \theta(1 - \{\log_2 n\}) \cdot n + 1$, with $\theta(x) := x - 2^x$ and $\{x\} := x - \lfloor x \rfloor$ is the *fractional part* of the real x. In particular, we have $0 \leqslant \{x\} < 1$. The derivative is $\theta'(x) = 1 - 2^x \log 2$; it has one root $\theta'(x_0) = 0 \Leftrightarrow x_0 = -\log_2 \log 2$ and it is positive before x_0, and negative after. Accordingly, $\theta(x)$ reaches its maximum at x_0: $\max_{0<x\leqslant 1} \theta(x) = \theta(x_0) = -(1 + \log\log 2)/\log 2 \simeq -0.9139$, and $\min_{0<x\leqslant 1} \theta(x) = \theta(1) = -1$. By injectivity, $\theta(1) = \theta(1 - \{\log_2 n\})$ implies $\{\log_2 n\} = 0$, that is, $n = 2^p$ (first case).

Hence $\mathcal{W}_n = n \log_2 n + A(\log_2 n) \cdot n + 1$, where $A(x) := 1 - \{x\} - 2^{1-\{x\}}$ is a periodic function, since $A(x) = A(\{x\})$, such that $-1 \leqslant A(x) < -0.91$. Further analysis of $A(x)$ requires Fourier series or complex analysis; its mean value is about -0.942695.

$$n \log_2 n - n + 1 \leqslant \mathcal{W}_n < n \log_2 n - 0.91n + 1. \tag{21}$$

The lower bound is attained when $n = 2^p$. The upper bound is most accurate when $\{\log_2 n\} = 1 + \log_2 \log 2$, that is, when n is the nearest integer to $2^p \log 2$ (take the binary expansion of $\log 2$, shift the point p times to the right and round). Of course, $\mathcal{W}_n \sim n \log_2 n$.

Average cost

Let $\mathcal{A}_n$ be the average number of comparisons to sort n keys top-down. All permutations of the input series being equally likely, equation (10) becomes

$$\mathcal{A}_0 = \mathcal{A}_1 = 0, \qquad \mathcal{A}_n = \mathcal{A}_{\lfloor n/2 \rfloor} + \mathcal{A}_{\lceil n/2 \rceil} + \mathcal{A}^{\bowtie}_{\lfloor n/2 \rfloor, \lceil n/2 \rceil},$$

which, with equation (5), in turn implies

$$\mathcal{A}_n = \mathcal{A}_{\lfloor n/2 \rfloor} + \mathcal{A}_{\lceil n/2 \rceil} + n - \frac{\lfloor n/2 \rfloor}{\lceil n/2 \rceil + 1} - \frac{\lceil n/2 \rceil}{\lfloor n/2 \rfloor + 1}. \tag{22}$$

Difference equations are not helpful here, so we should try an inductive approach instead, as we did for finding bounds on $\mathcal{B}_n$. Inequalities (9) are equivalent to $n \log_2 n - \alpha n + \alpha < \mathcal{A}_n < n \log_2 n - \alpha n + 2$ where $n = 2^p$, and this suggests us to also look for bounds of the form $n \log_2 n + an + b$ when $n \neq 2^p$.

Let us start with the lower bound and set to maximise the real constants a and b in

$$\mathsf{H}(n)\colon n \log_2 n + an + b \leqslant \mathcal{A}_n, \text{ for } n \geqslant 2.$$

Since $\mathsf{H}(2p)$ depends on $\mathsf{H}(p)$, and $\mathsf{H}(2p+1)$ depends on $\mathsf{H}(p)$ and $\mathsf{H}(p+1)$, the property $\mathsf{H}(n)$, for any $n > 1$, transitively depends on $\mathsf{H}(2)$ alone, because we are iterating divisions by 2. $\mathsf{H}(2)$ is equivalent to

$$2a + b + 1 \leqslant 0. \tag{23}$$

Because the definition of $\mathcal{A}_n$ depends on the parity of n, the inductive step will be twofold. Let us assume $\mathsf{H}(m)$ for $m < 2p$, in particular, we suppose $\mathsf{H}(p)$, which, with the expression of $\mathcal{A}_{2p}$ from equation (22), entails

$$(2p \log_2 p + 2ap + 2b) + 2p - 2 + \frac{2}{p+1} \leqslant \mathcal{A}_{2p}.$$

We want $\mathsf{H}(2p)$: $2p\log_2(2p) + 2ap + b = 2p\log_2 p + 2ap + 2p + b \leqslant \mathcal{A}_{2p}$, which holds if the following condition does:

$$2p\log_2 p + 2ap + 2p + b \leqslant 2p\log_2 p + 2ap + 2b + 2p - 2 + \frac{2}{p+1},$$

which is equivalent to

$$2 - \frac{2}{p+1} = \frac{2p}{p+1} \leqslant b.$$

Let $\Phi(p) := 2p/(p+1)$. This function is strictly increasing for $p > 0$ and $\Phi(p) \rightarrow 2^-$, as $p \rightarrow +\infty$.

The other inductive step deals with the odd values of n. We assume $\mathsf{H}(m)$ for all $m < 2p+1$, in particular, we suppose $\mathsf{H}(p)$ and $\mathsf{H}(p+1)$, which, with the expression of $\mathcal{A}_{2p+1}$ from equation (22), implies

$$(p\log_2 p + ap + b) + ((p+1)\log_2(p+1) + a(p+1) + b) + 2p - 1 + \frac{2}{p+2} \leqslant \mathcal{A}_{2p+1},$$

which may be simplified slightly into

$$p\log_2 p + (p+1)\log_2(p+1) + a(2p+1) + 2b + 2p - 1 + \frac{2}{p+2} \leqslant \mathcal{A}_{2p+1}.$$

We want to prove $\mathsf{H}(2p+1)$: $(2p+1)\log_2(2p+1) + a(2p+1) + b \leqslant \mathcal{A}_{2p+1}$, which is thus implied by

$$(2p+1)\log_2(2p+1) \leqslant p\log_2 p + (p+1)\log_2(p+1) + b + 2p - 1 + \frac{2}{p+2}. \tag{24}$$

Let $\Psi(p) := (2p+1)\log_2(2p+1) - (p+1)\log_2(p+1) - p\log_2 p - 2p + 1 - 2/(p+2)$. Then (24) is equivalent to $\Psi(p) \leqslant b$. Furthermore,

$$\frac{d\Psi}{dp}(p) = \frac{2}{(p+2)^2} + \log_2 \frac{4p^2+4p+1}{p^2+p} - 2 = \frac{2}{(p+2)^2} + \log_2\left(1 + \frac{1}{4p(p+1)}\right).$$

Clearly, $d\Psi/dp > 0$, for all $p > 0$, so $\Psi(p)$ is strictly increasing for $p > 0$. Let us find $\lim_{p\to+\infty}\Psi(p)$ by rewriting $\Psi(p)$ as follows:

$$\begin{aligned}\Psi(p) &= 2 - \frac{2}{p+2} + (2p+1)\log_2(p+\tfrac{1}{2}) - (p+1)\log_2(p+1) - p\log_2 p\\ &= 2 - \frac{2}{p+2} + p\left(\log_2(p+\tfrac{1}{2})^2 - \log_2(p+1) - \log_2 p\right) + \log_2(p+\tfrac{1}{2})\\ &\quad - \log_2(p+1)\\ &= 2 - \frac{2}{p+2} + p\log_2\left(1 + \frac{1}{4p(p+1)}\right) + \log_2\frac{p+1/2}{p+1}.\end{aligned}$$

The limit of $x\log(1+1/x^2)$ as $x \to +\infty$ can be found by changing x into $1/y$ and considering the limit as $y \to 0^+$, which is shown by l'Hôpital's rule to be 0. This result can be extended to apply to the large term in $\Psi(p)$ and, since all the other variable terms converge to 0, we can conclude that $\Psi(p) \to 2^-$, as $p \to +\infty$.

Because we need to satisfy the conditions $\Psi(p) \leqslant b$ and $\Phi(p) \leqslant b$ for both inductive steps to hold, we have to compare $\Psi(p)$ and $\Phi(p)$, when p is a natural number: we have $\Phi(1) < \Psi(1)$ and $\Phi(2) < \Psi(2)$, but $\Psi(p) < \Phi(p)$ if $p \geqslant 3$. Therefore, for b not to depend on p, we need it to be greater than 2, the smallest upper bound of Φ and Ψ. Inequality (23) means that we need to minimise b in order to maximise a (which is the priority), so we settle for the limit: $b_{\min} = 2$, and the same inequality entails $a \leqslant -3/2$, hence $a_{\max} = -3/2$. The principle of complete induction finally establishes that, for $n \geqslant 2$,

$$n\log_2 n - \frac{3}{2}n + 2 < \mathcal{A}_n. \tag{25}$$

This bound is not excellent, but it was not too hard to obtain. We may recall the lower bound when $n = 2^p$, in (9): $n\log_2 n - \alpha n + \alpha < \mathcal{A}_n$, where $\alpha \simeq 1.264499$. In fact, Flajolet and Golin [4] proved

$$n\log_2 n - \alpha n < \mathcal{A}_n. \tag{26}$$

Asymptotically, that bound is, up to the linear term, the same as for the case $n = 2^p$. Our inductive method cannot reach this nice result because it yields sufficient conditions that are too strong, in particular, we found no obvious way to get the decomposition $\mathcal{A}_{2^p+i} = \mathcal{A}_{2^p} + \mathcal{A}_i + \dots$

Now, let us find the smallest real constants a' and b' such that for $n \geqslant 2$, $\mathcal{A}_n \leqslant n\log_2 n + a'n + b'$. The base case of $H(n)$ in (23) is here reversed: $2a' + b' + 1 \geqslant 0$. Hence, in order to minimise a', we need to maximise b'. Furthermore, the conditions on b' from the inductive steps are reversed as well with respect to b: $b' \leqslant \Phi(p)$ and $b' \leqslant \Psi(p)$. The base case is $H(2)$, that is, $p = 1$, and we saw earlier that $\Phi(1) \leqslant \Psi(1)$, thus we must have $b' \leqslant \Phi(1) = 1$. The maximum value is thus $b'_{\text{max}} = 1$. Finally, this implies that $a' \geqslant -1$, thus $a'_{\text{min}} = -1$.

Gathering the bounds, we hence established that

$$n \log_2 n - \frac{3}{2}n + 2 < \mathcal{A}_n < n \log_2 n - n + 1.$$

Trivially, we have $\mathcal{A}_n \sim n \log_2 n \sim \mathcal{W}_n \sim 2 \cdot \mathcal{B}_n$. Flajolet and Golin [4] proved, using complex analysis the following very strong result:

$$\mathcal{A}_n = n \log_2 n + B(\log_2 n) \cdot n + O(1),$$

where B is continuous, non-differentiable, periodic with period 1, of mean value -1.2481520. The notation $O(1)$ is an instance of Bachmann's notation for an unknown positive constant. The maximum of $B(x)$ is approximately -1.24075, so

$$\mathcal{A}_n = n \log_2 n - (1.25 \pm 0.01) \cdot n + O(1).$$

4. Conclusion

We have shown that it is possible to obtain good bounds on the minimum, maximum and average number of comparisons of merge sort without resorting to advanced real analysis, like Fourier analysis, nor complex analysis, like Mellin transforms. Whilst these powerful techniques indeed bring the best results, they are not suitable for postgraduate students in informatics who are introduced to sorting algorithms.

Bachmann's O notation is often misused for lower bounds and it is deceptively simple because it must be abused to be really useful and it seems to conflict with algebra over numbers. Using the Θ notation when possible is better, but the extra work of doing so then may become on par with determining asymptotic equivalences, where multiplicative constants are explicit. It is probably an overkill for most students to find out the exact linear coefficients, but, for the most motivated, this paper shows how to do so at the cost of slightly less precision sometimes.

Acknowledgements

The author thanks the anonymous reviewers for their helpful comments on the manuscript.

References

[1] D. E. Knuth, Selected Papers on Computer Science, in: *Von Neumann's First Computer Program*, CSLI Publications, 1996, 205–226.

[2] R. L. Graham, D. E. Knuth and O. Patashnik, *Concrete Mathematics*, 2nd edition, Addison-Wesley, 1994.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd edition, The MIT Press, 2009.

[4] P. Flajolet and M. Golin, Mellin Transforms and Asymptotics: The Mergesort Recurrence, *Acta Informatica* **31**, no. 7 (1994), 673–696.

[5] H.-K. Hwang, Asymptotic expansions of the mergesort recurrences, *Acta Informatica* **35**, no. 11 (1998), 911–919.

[6] W.-M. Chen, H.-K. Hwang and G.-H. Chen, The cost distribution of queue-mergesort, optimal mergesorts, and power-of-2 rules, *Journal of Algorithms* **30**, no. 2 (1999), 423–448.

[7] D. E. Knuth, *Sorting and Searching*, in: *The Art of Computer Programming*, Vol. 3, 2nd edition, Addison-Wesley, 1998.

[8] M. J. Golin and R. Sedgewick, Queue-mergesort, *Information Processing Letters* **48**, no. 5 (1993), 253–259.

[9] D. E. Knuth, *Selected Papers on the Analysis of Algorithms*, in: *Big Omicron and Big Omega and Big Theta*, CSLI Publications, 2000, 35–41.

[10] P. B. Borwein, *On the irrationality of certain series*, in: *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 112, 1992, 141–146.

[11] M. D. McIlroy, The number of 1's in the binary integers: Bounds and extremal properties, *SIAM Journal on Computing* **3**, no. 4 (1974), 255–261, Society for Industrial and Applied Mathematics.

[12] L. E. Bush, An Asymptotic Formula for the Average Sum of the Digits of Integers, *The American Mathematical Monthly* **47**, no. 3 (1940), 154–156.

[13] H. Delange, Sur la fonction sommatoire de la fonction "somme des chiffres", *L'Enseignement Mathématique* **XXI**, no. 1 (1975), 31–47.

[14] D. E. Knuth, *Fundamental Algorithms*, in: *The Art of Computer Programming*, Vol. 1, 3rd edition, Addison-Wesley, 1997.

CHRISTIAN RINDERKNECHT
DEPARTMENT OF PROGRAMMING LANGUAGES AND COMPILERS
EÖTVÖS LORÁND UNIVERSITY
BUDAPEST, HUNGARY

E-mail: `rinderkn@caesar.elte.hu`

(Received November, 2012)