

Article

Elementary yet Precise Best-Case Analysis of MergeSort with an Application to the Sum of Digits Problem

Marek A. Suchenek 

Department of Computer Science, California State University, Dominguez Hills, 1000 E. Victoria St.,
Carson, CA 90747, USA; suchenek@csudh.edu

Abstract

An exact formula $B(n) = \frac{n}{2}(\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}(\frac{n}{2^{k+1}})$, where $\text{Zigzag}(x) = \min(x - \lfloor x \rfloor, \lceil x \rceil - x)$, for the minimum number $B(n)$ of comparisons of keys performed by MergeSort on an n -element array is derived and analyzed. The said formula is less complex than any other known formula for the same and can be evaluated in $O(\log^c)$ time, where c is a constant. It is shown that there is no closed-form formula for the above. Other variants for $B(n)$ are described as well. Since the recurrence relation for the minimum number of comparisons of keys for MergeSort is identical with a recurrence relation for the number of 1s in binary expansions of all integers between 0 and n (exclusively), the above results extend to the sum of binary digits problem.

Keywords: MergeSort; sum of digits; sorting; best case

MSC: 11A63; 11B37; 68P10; 68R99; 68W40

1. Introduction

“One Picture is Worth a Thousand Words”

[An advertisement for the *San Antonio Light* (1918)]

Teaching undergraduate Analysis of Algorithms has been a rewarding, although a bit taxing, experience. I was often surprised to learn that many basic problems that clearly belong to its core syllabus had been left unanswered or partially answered. Additionally, it seemed a bit odd to me that many otherwise decent texts offered unnecessarily imprecise computations (text [1] being a notable exception in this category) of several rather fundamental results.

In this article, I pursue a seemingly marginal—albeit fundamental—topic, a precise characterization of the best-case behavior of a well-known sorting algorithm, MergeSort, whose pursuit, however, yields some interesting findings that could hardly be characterized as “marginal.” It turns out that—contrary to what a casual student of this subject might believe—computing the exact formula for the number of comparisons of keys that MergeSort performs on any n -element array in the best case is not a routine exercise (in fact, it is significantly more complicated than precise derivation of the well-known exact formula for the number of comparisons of keys performed by it in the worst case) and leads to an instance of a problem that gained some notoriety for being a hard nut to crack analytically: the *sum of digits* problem. Even more unexpectedly, the relatively straightforward (although not quite *closed-form*) formula for the said number of comparisons yields an improvement of a well-known answer to this instance of the *sum of digits* problem:



Academic Editor: Marjan Mernik

Received: 4 January 2026

Revised: 31 January 2026

Accepted: 9 February 2026

Published: 21 February 2026

Copyright: © 2026 by the Author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

How many 1s appear in binary representations of all integers between (but not including) 0 and n ?

2. Materials and Methods

Several derivations presented in this article may be characterized as experimental mathematics. Once they produced experimentally (using Wolfram Mathematica software) the desired (sought-after) formulas, finding analytic proofs of the said formulas became a relatively easy and straightforward exercise, and has been done in Sections 9–11. Those derivations provide an insightful exercise that demonstrates how certain, apparently, hard-to-solve recurrence relations pertaining to the complexity of divide-and-conquer algorithms can be practically and precisely solved by means of experiments and computations augmented with formal analytic proofs of the derived solutions.

No materials were used in the research described in this article.

3. Results

The main result of this article is the following formula of the [Main Theorem](#) that gives the exact number $B(n)$ of comparisons of keys performed by the classic MergeSort algorithm in the best case while sorting an n -element array.

$$B(n) = \frac{n}{2}(\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}\left(\frac{n}{2^{k+1}}\right),$$

where

$$\text{Zigzag}(x) = \min(x - \lfloor x \rfloor, \lceil x \rceil - x).$$

The above formula may be evaluated in sublinear $O(\log^c n)$ time, where c is a constant, that is considerably shorter than the time needed for running MergeSort on its best-case input array (for instance, an array of consecutive integers) of n elements or for counting directly how many times digit 1 occurs in binary representations of all integers between 1 and n . This constitutes a significant improvement over other exact formula, attributed to J. R. Trollope and discussed in Section 8, for the said number of 1s that involves infinite summation. Even if only finitely many terms in that infinite summation are non-zero, the Trollope formula does not indicate how long the process of its computation will take for any value of n . For instance, the Trollope formula may possibly require more time to evaluate than counting directly how many times digit 1 occurs in binary representations of all integers between 1 and n . (See Note 1 at the end of this Section.)

The exactness of the formula derived and proved in this article allows for its definite experimental verification, which in general cannot be done for characterizations given by Big-O/Big-Ω/Big-Θ notation. Such an experimental verification has been performed by the author using a Java program that incorporates the code for Merge visualized in Figure 1.

Moreover, the exact and practical way to compute the answer to the question of how many comparisons of key does MergeSort perform in the best case while sorting an n -element array fills a significant loophole in many computer science faculty and students' knowledge of the time complexity of MergeSort and satisfies the cognitive curiosity of those who realize the advantages of knowing things *exactly* in exact sciences and mathematics. (See Note 2 at the end of this Section.)

It is also demonstrated (see Corollary 3) that there is no exact closed-form formula that gives the said number of comps and (in Section 7) that the said number is significantly larger (by a term of size $\Theta(n)$) than half of the number of comps performed by MergeSort in the worst case.

```

70
71 private static int[] Merge(int[] A, int[] B)
72     // Merges two sorted arrays into one sorted array
73 {
74     int[] C = new int[A.length + B.length];
75     int indexA = 0, indexB = 0, indexC = 0;
76     while ((indexA < A.length) && (indexB < B.length))
77     {
78         if (A[indexA] < B[indexB] & Bcnt.incr()) // move A[indexA] to C
79             C[indexC++] = A[indexA++];
80         else // move B[indexA] to B
81             C[indexC++] = B[indexB++];
82     }
83     // copy the remaining part of A or B to C
84     if (indexA < A.length) // copy the remaining part of A
85         for (int i = indexA; i < A.length; i++)
86             C[indexC++] = A[i];
87     else // copy the remaining part of B
88         for (int i = indexB; i < B.length; i++)
89             C[indexC++] = B[i];
90     // cnt2.incr();
91     return C;
92 }
93

```

Figure 1. A Java code of Merge, based on a pseudo-code from [2]. Calls to Boolean method Bcnt.incr() count the number of comps and facilitate experimental verification of the formula derived and proved in this article.

Note 1. Since the total number of significant bits in binary representations of all numbers between 1 and n is equal to $\sum_{i=1}^n (\lfloor \lg i \rfloor + 1)$, the direct counting of 1s in all those numbers requires at least $\lg n!$ or, by virtue of Stirling's formula, $n \lg n - n \lg e + O(\lg n)$ steps that take $\Theta(n \log n)$ time to carry them on, which is significantly more than the $O(\log^c n)$ time that is sufficient for the evaluation of Formula (15) derived and proved in this article.

Note 2. A student of mine was once trying to convince me that the number of comparisons of keys that MergeSort performs in the worst case was exactly equal to twice the number it performed in the best case. In order to show her why it was not exactly the case, I experimentally derived an exact formula for it. This is how I got into writing this article.

4. MergeSort and Its Best-Case Behavior

A call to MergeSort inherits an n -element array A of integers and sorts it non-decreasingly, following the steps described below.

A Java code of Merge is shown in Figure 1.

A typical measure of the running time of MergeSort is the number of comparisons of keys, which for brevity I call *comps*, that it performs while sorting array A . Since no comps are performed outside Merge, the running time of MergeSort can be computed as the sum of numbers of comps performed by all calls to Merge during the execution of MergeSort.

Because all comps during the execution of Merge are performed within the while-loop that begins in line 76 of the code shown in Figure 1, and the said loop does not terminate before the last element of one of the two arrays has been compared with an element of the other array, the number of comps performed by Merge on two arrays is never less than the size of any of the two arrays passed to it, which in the case of step 2c of Algorithm 1 is not less than $\lfloor \frac{n}{2} \rfloor$, thus making $\lfloor \frac{n}{2} \rfloor$ a lower bound for the said number of comps. Since the number of comps performed by Merge in the case where all the elements of the first array of size $\lfloor \frac{n}{2} \rfloor$ are less than all the elements of the second array of size $\lceil \frac{n}{2} \rceil$ is actually equal to the size of the first array, it follows that $\lfloor \frac{n}{2} \rfloor$ is the minimum number of comps performed during the execution of Merge (and not just a lower bound of that number).

Algorithm 1 MergeSort

To sort an n -element array A , do the following:

1. If $n \leq 1$, then return A to the caller;
2. If $n \geq 2$, then
 - (a) Pass the first $\lfloor \frac{n}{2} \rfloor$ elements of A to a recursive call to MergeSort;
 - (b) Pass the last $\lceil \frac{n}{2} \rceil$ elements of A to another recursive call to MergeSort;
 - (c) Linearly merge, by means of a call to Merge, the non-decreasingly sorted arrays that were returned from those calls onto one non-decreasingly sorted array A' ;
 - (d) Return A' to the caller.

Obviously, increasingly sorted array A of any size $n \geq 2$ produces only best-case scenarios for all subsequent calls to Merge. Therefore, $\lfloor \frac{k}{2} \rfloor$ —where $k \geq 2$ is the size of any sub-array A of size greater than or equal to 2 passed to any (recursive) call to MergeSort—is the actual minimum of and not just a lower bound on the said number of comps performed by Merge invoked by the said call to MergeSort. This fact allows for a rudimentary analysis of the recursion tree for MergeSort that easily yields—as we will see in the next two subsections—the exact Formula (7) for the minimum number of comps for the entire MergeSort. Without said fact, one could only derive by means of analysis of the recursion tree a lower bound on the number of comps performed by MergeSort, even though—as I have pointed out, above—the size $\lfloor \frac{n}{2} \rfloor$ of the first sub-array passed to Merge is the actual minimum of comps performed at this level and not just a lower bound.

The problem arises when one tries to reduce the said formula, which naturally involves long summations, to one that can be evaluated in a logarithmic time.

4.1. Recursion Tree

The obvious recursion tree for MergeSort and sufficiently large n is shown in Figure 2 and explained in some detail in the caption below it.

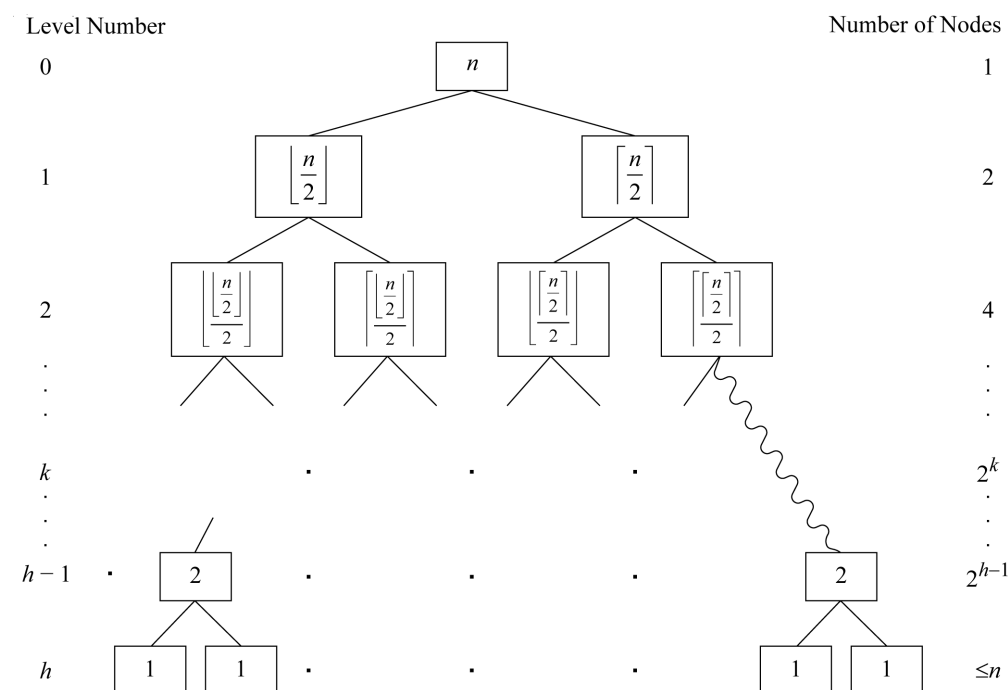


Figure 2. A sketch of the recursion 2-tree T for MergeSort for $n \geq 10$, with level numbers shown on the left and the numbers of nodes in respective levels shown on the right. The nodes correspond to calls to MergeSort and show the sizes of (sub)arrays passed to those calls. The last level number is h ,

the depth of T ; it only contains nodes with the value 1. The root corresponds to the original call to MergeSort. If a call that is represented by a node p executes further recursive calls to MergeSort, then these calls are represented by the children of p ; otherwise, p is a leaf. Thus a node in T is a leaf if, and only if, it shows the number 1. The level $h - 1$ of tree T may or may not contain some (but not all) leaves (not shown on the sketch), and if it does contain leaves then the level h of tree T contains less than n nodes, all of them being leaves. In any case, the total number of leaves in tree T is n . The wavy line ~~~~~ represents a path of the right child in T .

A recursive application of the equality

$$\lceil \frac{n}{2} \rceil = \lfloor \frac{n+1}{2} \rfloor \quad (1)$$

(which can be easily verified separately for odd and even values of n) allows for rewriting of that tree onto one whose first four levels are shown in Figure 3.

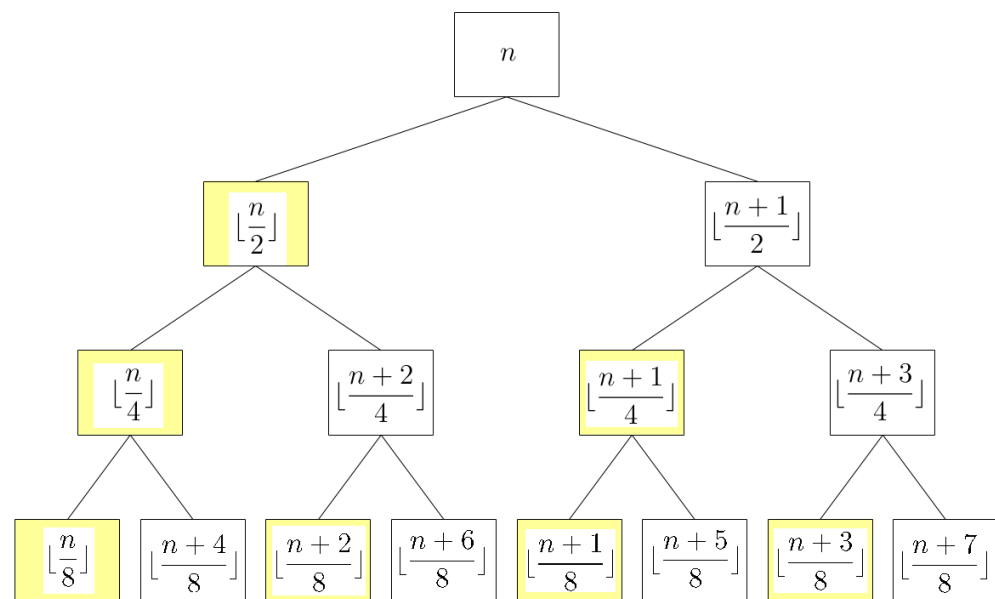


Figure 3. The first four levels of the recursion 2-tree T from Figure 2, with the equality (1) applied, recursively. The number of comparisons of keys performed in the best case by Merge invoked in step 2c of Algorithm 1 as a result to a call to MergeSort corresponding to a node of T is equal to the number that is shown in its left child, highlighted yellow. All the children of the nodes at any level $k \leq h - 1$ of T show numbers of the form $\lfloor \frac{n+i}{2^{k+1}} \rfloor$, where $0 \leq i < 2^{k+1}$. All the right children of the nodes at the level k of T show numbers of the form $\lfloor \frac{n+i}{2^{k+1}} \rfloor$, where $2^k \leq i < 2^{k+1}$. Thus all the left children (highlighted yellow) at the level k show numbers of the form $\lfloor \frac{n+i}{2^{k+1}} \rfloor$, where $0 \leq i < 2^k$.

4.2. Recurrence Relation $B(n)$ for the Minimum Numbers of Comps

As I have noticed, before, the number of comps performed by a single call to MergeSort in the best case on an n -element array for $n \geq 2$ is equal to $\lfloor \frac{n}{2} \rfloor$. (Recall that all comps performed during a single call to MergeSort are done within the call to Merge that MergeSort invokes after both recursive calls to itself have been completed.)

Therefore, the following recurrence relation for the minimum number $B(n)$ of comparisons of keys that MergeSort performs on any n -element array is straightforward to derive from its description given by Algorithm 1, as follows:

$$B(1) = 0, \quad (2)$$

and for $n \geq 2$,

$$B(n) = \lfloor \frac{n}{2} \rfloor + B(\lfloor \frac{n}{2} \rfloor) + B(\lceil \frac{n}{2} \rceil). \quad (3)$$

Using the equality (1), the recurrence relation (3) is equivalent to

$$B(n) = \lfloor \frac{n}{2} \rfloor + B(\lfloor \frac{n}{2} \rfloor) + B(\lfloor \frac{n+1}{2} \rfloor). \quad (4)$$

A graph of $B(n)$ is shown in Figure 4.

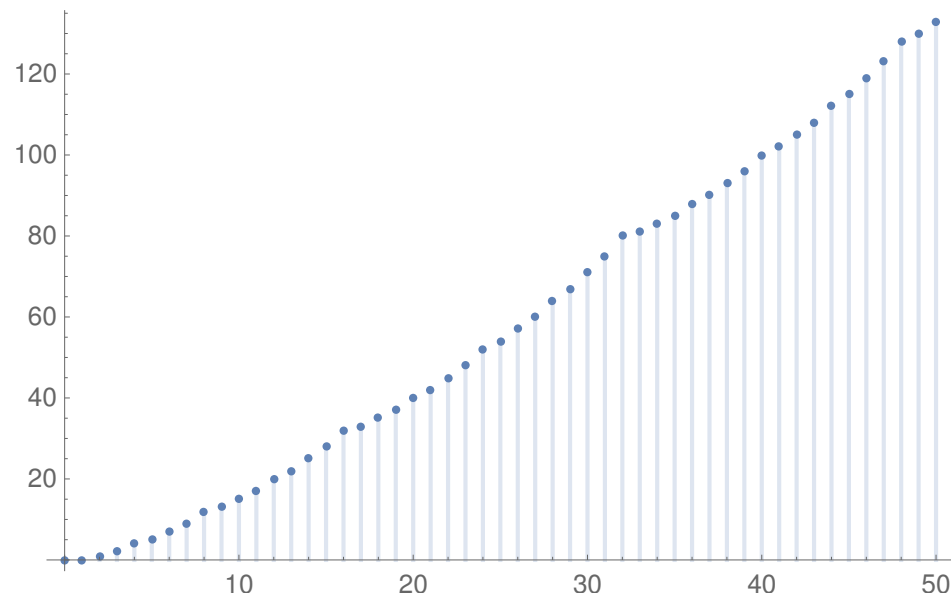


Figure 4. Graph of the solution $B(n)$ of the recurrences (2) and (4).

Note 3. If n is a power of 2, that is, if $n = 2^{\lceil \lg n \rceil}$, then

$$B(n) = \frac{n}{2} \lg n.$$

Indeed, in such a case, recurrences (3) and (4) simplify to $B(n) = \frac{n}{2} + 2B(\frac{n}{2})$, so that $B(n) = 2\frac{n}{2} + 2B(\frac{n}{2}) = \dots = \lceil \lg n \rceil \frac{n}{2} + 2B(\frac{n}{2^{\lceil \lg n \rceil}}) = \lceil \lg n \rceil \frac{n}{2} + 2B(1) = [\text{since } \lceil \lg n \rceil = \lg n \text{ and, by the equality (2), } B(1) = 0] = \frac{n}{2} \lg n.$

4.3. A Solution of Recurrence Relation $B(n)$

Unfolding the recurrence (4) allows for noticing that the minimum number $B(n)$ of comps performed by all calls to Merge is equal to the sum of all values shown at nodes highlighted yellow in the recursion tree T of Figure 3. They may be summed up level by level. One can notice from Figure 3 that the number of comps performed at any level k with the maximal number 2^k of nodes is given by this formula:

$$\sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor. \quad (5)$$

What is not clear is whether all levels of the recursion tree T are maximal. Fortunately, the answer to this question does not depend on whether a given instance of MergeSort is running on a best-case array or on any other case of array. It has been known from a classic analysis of the worst-case running time of MergeSort that every level of its recursion tree T that contains at least one non-leaf, or—in other words—a node that shows the value $p \geq 2$, is maximal. Appendix A, contains a detailed derivation of that fact. Thus all levels

0 through $h - 1$ of T are maximal. Therefore, Formula (5) gives the number of comps for every level $0 \leq k \leq h - 1$.

The last level h of T may be not maximal because the level $h - 1$ may contain leaves, or—in other words—nodes that show the value $p = 1$, where $p = \lfloor \frac{n+i}{2^{h-1}} \rfloor$ for some $0 \leq i \leq 2^{h-1} - 1$, and as such do not have any children in the level h . However, for each such node, the value of $\lfloor \frac{p}{2} \rfloor = \lfloor \frac{n+i}{2^h} \rfloor$ is 0, so it can be included in summation (5) without affecting its value even though the said value does not correspond to any node in the level h . Therefore, Formula (5) gives the number of comps for the level $k = h$.

Additionally, the depth h of T is known to be equal to $\lceil \lg n \rceil$, as Theorem A2 in Appendix A states. Thus the minimum number of comps performed by MergeSort is given by this summation

$$\sum_{k=0}^{\lceil \lg n \rceil} \sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor. \quad (6)$$

Since $n - 1 < 2^{\lceil \lg n \rceil}$ so that $n + 2^{\lceil \lg n \rceil} - 1 < 2^{\lceil \lg n \rceil + 1}$ and for every $0 \leq i \leq 2^{\lceil \lg n \rceil} - 1$, $n + i < 2^{\lceil \lg n \rceil + 1}$, thus making

$$\lfloor \frac{n+i}{2^{\lceil \lg n \rceil + 1}} \rfloor = 0,$$

it follows that the value of the inner summation (5) in summation (6) for $k = h = \lceil \lg n \rceil$ is 0. Therefore, Formula (6) is equal to this slightly shorter formula:

$$\sum_{k=0}^{\lceil \lg n \rceil} \sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor. \quad (7)$$

Unfortunately, summation (7) contains $n - 1$ non-zero terms (each corresponding to one of $n - 1$ internal nodes of 2-tree T), so it cannot be evaluated quickly in its present form. Fortunately, as I am going to demonstrate, its inner summation (5) can be reduced to a closed-form formula (13).

4.4. Zigzag Function

In order to reduce (5) to a closed form, I am going to use the function *Zigzag* defined by

$$\text{Zigzag}(x) = \min(x - \lfloor x \rfloor, \lceil x \rceil - x). \quad (8)$$

The following fact is instrumental for that purpose.

Theorem 1. For every natural number n and every positive natural number m ,

$$\sum_{i=m}^{2m-1} \lfloor \frac{n+i}{2m} \rfloor - \sum_{i=0}^{m-1} \lfloor \frac{n+i}{2m} \rfloor = 2m \times \text{Zigzag}\left(\frac{n}{2m}\right), \quad (9)$$

where *Zigzag* is a function defined by (8) and visualized in Figure 5.

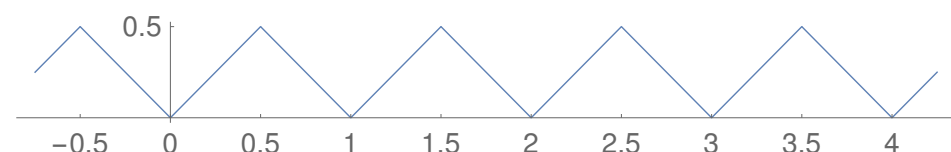


Figure 5. Graph of function $\text{Zigzag}(x) = \min(x - \lfloor x \rfloor, \lceil x \rceil - x)$.

Proof. The equality (9) can be verified experimentally, for instance, with the help of software for symbolic computation; I used Wolfram Mathematica for that purpose. The analytic proof is deferred to Section 9. $\square$

Corollary 1. For every natural number n and every positive natural number m ,

$$\sum_{i=0}^{m-1} \lfloor \frac{n+i}{2m} \rfloor = \frac{n}{2} - m \times \text{Zigzag}(\frac{n}{2m}), \quad (10)$$

where *Zigzag* is a function defined by (8) and visualized in Figure 5.

Proof. First, let us note (analytic proof of this fact is a straightforward exercise; see Appendix B) that

$$\sum_{i=0}^{2m-1} \lfloor \frac{n+i}{2m} \rfloor = n. \quad (11)$$

From (11), I conclude

$$\sum_{i=0}^{m-1} \lfloor \frac{n+i}{2m} \rfloor + \sum_{i=m}^{2m-1} \lfloor \frac{n+i}{2m} \rfloor = n. \quad (12)$$

Solving Equations (9) and (12) for $\sum_{i=0}^{m-1} \lfloor \frac{n+i}{2m} \rfloor$ yields (10). $\square$

Here is the closed form of summation (5).

Corollary 2. For every natural number n and every natural number k ,

$$\sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor = \frac{n}{2} - 2^k \text{Zigzag}(\frac{n}{2^{k+1}}), \quad (13)$$

where *Zigzag* is a function defined by (8) and visualized in Figure 5.

Proof. Substitute $m = 2^k$ in (10). $\square$

The following theorem yields Formula (14) for the minimum number $B(n)$ of comps performed by MergeSort.

Theorem 2. For every natural number n ,

$$\sum_{k=0}^{\lfloor \lg n \rfloor} \sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor = \frac{n}{2} (\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}(\frac{n}{2^{k+1}}), \quad (14)$$

where *Zigzag* is a function defined by (8) and visualized in Figure 5.

Proof.

$$\begin{aligned} \sum_{k=0}^{\lfloor \lg n \rfloor} \sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor &= \sum_{k=0}^{\lfloor \lg n \rfloor} (\frac{n}{2} - 2^k \text{Zigzag}(\frac{n}{2^{k+1}})) = \\ &= \frac{n}{2} (\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}(\frac{n}{2^{k+1}}). \end{aligned}$$

$\square$

Formula (14), although not a quite closed form, comprises summation with only $\lfloor \lg n \rfloor + 1$ closed-form terms, so it may be evaluated in $O(\log^c)$ time, where c is a constant. I will show in Section 5 that (14) does not have a closed form. Graphs of both sides of equality (14) are shown in Figure 6. One can see that, for natural numbers n , they coincide with the solution $B(n)$ of recurrences (2) and (3) visualized in Figure 4.

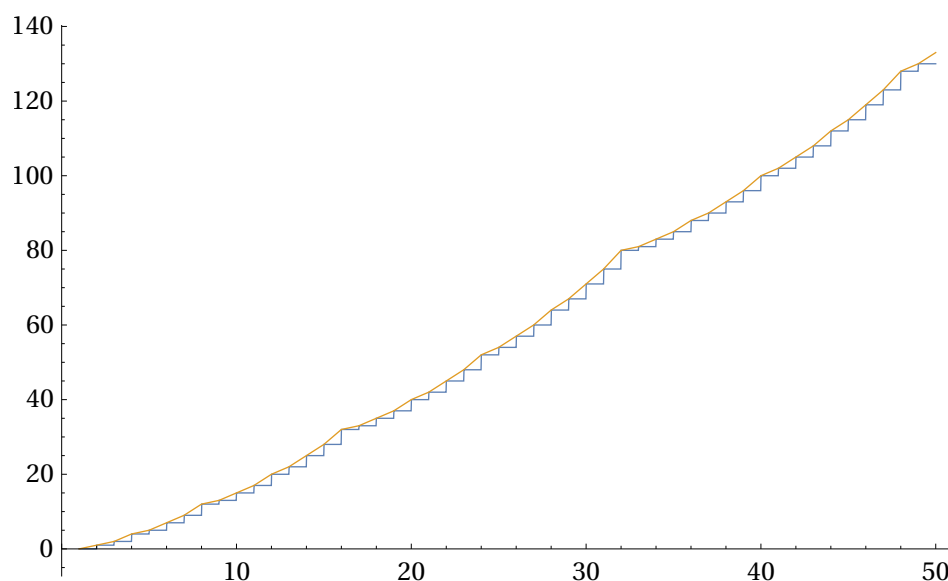


Figure 6. Graphs of functions $\sum_{k=0}^{\lfloor \lg n \rfloor} \sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor$ (bottom line) and $\frac{n}{2}(\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}(\frac{n}{2^{k+1}})$ (top line) of equality (14). They coincide with each other for all natural numbers n .

Main Theorem. For every natural number n , the minimum number $B(n)$ of comps that MergeSort performs while sorting an n -element array is

$$B(n) = \frac{n}{2}(\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}\left(\frac{n}{2^{k+1}}\right), \quad (15)$$

where Zigzag is a function defined by (8) and visualized in Figure 5.

Proof. As it has been shown in Section 4.3, Formula (7) gives the minimum number $B(n)$ of comps performed by MergeSort on any n -element array. Thus substituting $B(n)$ for

$$\sum_{k=0}^{\lfloor \lg n \rfloor} \sum_{i=0}^{2^k-1} \lfloor \frac{n+i}{2^{k+1}} \rfloor$$

in equality (14) of Theorem 2 yields (15). $\square$

5. A Fractal in $B(n)$

A deceitfully simple expression

$$\sum_{k=0}^{\lfloor \lg x \rfloor} 2^{k+1} \text{Zigzag}\left(\frac{x}{2^{k+1}}\right), \quad (16)$$

half of which occurs in Formula (15) of the **Main Theorem**, is a formidable adversary for those who may try to turn it into a closed form, although the time required for its evaluation for any given n is $O(\log^c)$ (so, for all practical purposes, some may consider (15) a closed-form formula). That does not come as a surprise, taking into account that its graph, shown in Figure 7, bears a resemblance of fractal. This can be easily seen as soon as a sawtooth function $2^{\lfloor \lg x \rfloor + 1} - x$ is subtracted from it, yielding the function $F(x)$ given by

$$F(x) = \sum_{k=0}^{\lfloor \lg x \rfloor} 2^{k+1} \text{Zigzag}\left(\frac{x}{2^{k+1}}\right) - 2^{\lfloor \lg x \rfloor + 1} + x. \quad (17)$$

Since $\frac{1}{2} \leq \frac{x}{2^{\lfloor \lg x \rfloor + 1}} < 1$, equality (8) implies

$$\text{Zigzag}\left(\frac{x}{2^{\lfloor \lg x \rfloor + 1}}\right) = 1 - \frac{x}{2^{\lfloor \lg x \rfloor + 1}},$$

or

$$2^{\lfloor \lg x \rfloor + 1} \text{Zigzag}\left(\frac{x}{2^{\lfloor \lg x \rfloor + 1}}\right) = 2^{\lfloor \lg x \rfloor + 1} - x. \quad (18)$$

Equality (18) simplifies definition (17) of function F to

$$F(x) = \sum_{k=1}^{\lfloor \lg x \rfloor} 2^k \text{Zigzag}\left(\frac{x}{2^k}\right), \quad (19)$$

visualized in Figure 8.

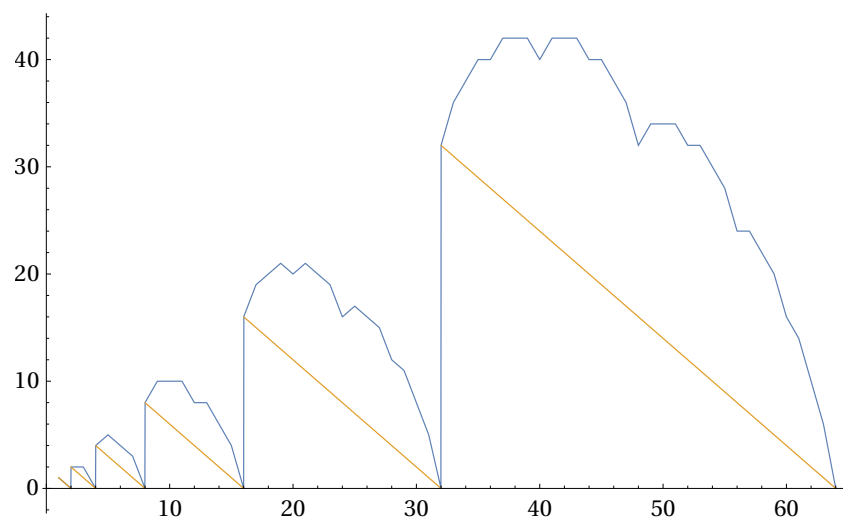


Figure 7. A graph of function $\sum_{k=0}^{\lfloor \lg x \rfloor} 2^{k+1} \text{Zigzag}\left(\frac{x}{2^{k+1}}\right)$ plotted against a sawtooth function $2^{\lfloor \lg x \rfloor + 1} - x$ (the orange line).

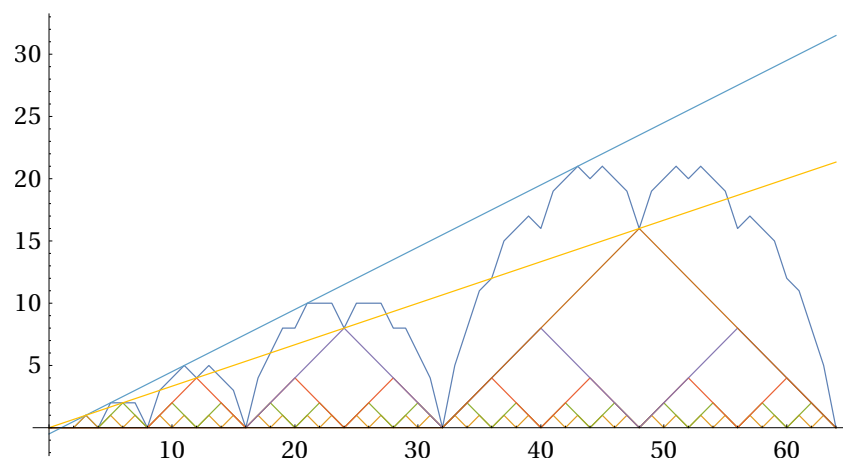


Figure 8. A graph of function $F(x) = \sum_{k=1}^{\lfloor \lg x \rfloor} 2^k \text{Zigzag}\left(\frac{x}{2^k}\right)$ plotted below its tight linear upper bound $y = \frac{x-1}{2}$ (it can be shown that $F(x) = \frac{x-1}{2}$ whenever $x = \frac{1}{3}(2^{k+1} + (-1)^k)$ for some integer $k \geq 0$); also shown below $F(x)$ are the terms $2^k \text{Zigzag}\left(\frac{x}{2^k}\right)$ (the brown lines) of the summation and their tight linear upper bound $y = \frac{x}{3}$ (the orange line).

The function F is a fractal with quasi similarity that repeats at intervals of exponentially growing length. It is a union

$$F = \bigcup_{k=0}^{\infty} f_k \quad (20)$$

of functions f_k , each having an interval $[2^k, 2^{k+1})$ as its domain. In other words, for every integer $k \geq 0$,

$$f_k = F \upharpoonright [2^k, 2^{k+1}), \quad (21)$$

which, of course, yields (20).

Let $\hat{f}_k$ be the normalized f_k on interval $[0, 1)$, defined by

$$\hat{f}_k(x) = \frac{1}{2^k} f_k(2^k(x+1)), \quad (22)$$

and $\tilde{f}_k$ be the periodized $\hat{f}_k$ by composing it with a sawtooth function $x - \lfloor x \rfloor$ (the fractional part of x) defined by

$$\tilde{f}_k(x) = \hat{f}_k(x - \lfloor x \rfloor). \quad (23)$$

Contracting definitions (21), (22), and (23) yields

$$\tilde{f}_k(x) = \frac{1}{2^k} F(2^k(x - \lfloor x \rfloor + 1)). \quad (24)$$

One can compute (an elementary geometric argument based on the graph visualized in Figure 9 will do) from (24) the following alternative formula for $\tilde{f}_k(x)$:

$$\tilde{f}_k(x) = \sum_{i=0}^{k-1} \frac{1}{2^i} \text{Zigzag}(2^i x). \quad (25)$$

Figure 9 shows functions $\tilde{f}_0, \dots, \tilde{f}_6$ drawn on the same graph.

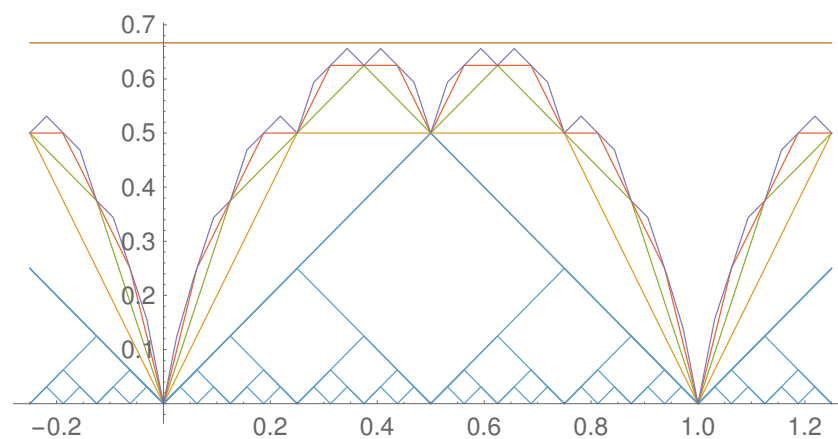


Figure 9. A graph of the first six (the first one is 0) normalized parts (in blue, orange, green, red and purple) of function F of Figure 8 plotted against the line $y = \sum_{i=0}^{\infty} \frac{1}{2^{2^i+1}} = \frac{2}{3}$ (in brown). Also shown (in blue) are the first five terms $\frac{1}{2^i} \text{Zigzag}(2^i x)$, $i = 0, \dots, 4$, of sums that occur in Formula (25) for $\tilde{f}_k(x)$; for each integer n and all $x \in [n, n+1)$, their parts above the X -axis restricted to $[n, n+1)$ visualize a fragment of an infinite binary search *trie* T defined as the set of shortest binary representations of $x - \lfloor x \rfloor$ with the last digit 1 (if the said binary representation is finite) being interpreted as the sequence terminator; in particular, the root of T is 0.1, and if $\mathbf{a}$ is a finite binary sequence, then the children of the binary representation 0. $\mathbf{a}$ 1 are 0. $\mathbf{a}$ 01 and 0. $\mathbf{a}$ 11.

Since each function f_k , and—therefore—each function $\hat{f}_k$, and—therefore—each function $\tilde{f}_k$, are a result of smaller and smaller triangles piled, originating in the function *Zigzag*

of definition (19) of function F , on one another as shown in Figure 9, for any integers $0 \leq i < j$, $\tilde{f}_i$ linearly interpolates $\tilde{f}_j$. Because of that, each $\tilde{f}_i$ linearly interpolates the limit $\tilde{F}$ of all $\tilde{f}_k$ s defined by

$$\tilde{F}(x) = \lim_{k \rightarrow \infty} \tilde{f}_k(x), \quad (26)$$

as Figure 10 illustrates. An application of (25) to (26) yields

$$\tilde{F}(x) = \sum_{i=0}^{\infty} \frac{1}{2^i} \text{Zigzag}(2^i x). \quad (27)$$

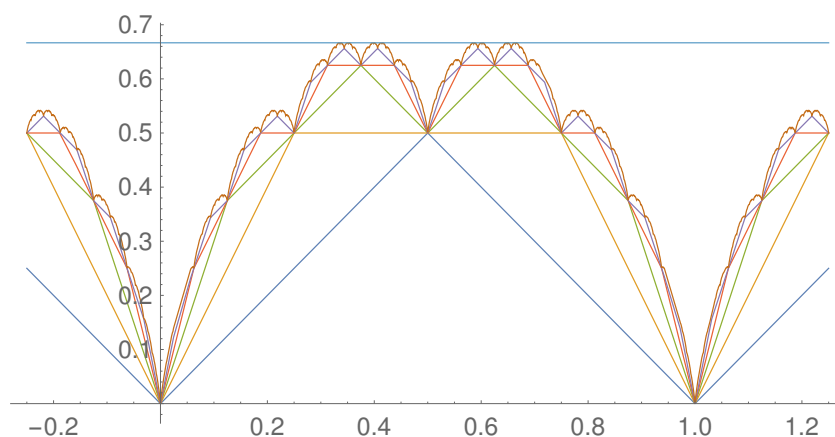


Figure 10. Functions $\tilde{f}_0(x), \tilde{f}_1(x), \dots$ and their limit (the topmost curve) $\tilde{F}(x)$ (each plotted in a different color) given by (27). Collapsing $\text{Zigzag}(x)$ would yield the same, albeit the scaled-down (by a factor of 2) pattern $\frac{1}{2}\tilde{F}(2x)$, as (27) does imply.

Since, for every integer n, k and $i \geq k$, $2^i \frac{n}{2^k}$ is integer, $\text{Zigzag}(2^i \frac{n}{2^k}) = 0$. Therefore, by virtue of (25) and (27), for every non-negative integer k and n ,

$$\tilde{F}\left(\frac{n}{2^k}\right) = \tilde{f}_k\left(\frac{n}{2^k}\right). \quad (28)$$

This and (25) eliminate the need for infinite summation (as it appears in (27)) while computing $\tilde{F}\left(\frac{n}{2^k}\right)$.

It can be shown that, although a continuous function, $\tilde{F}$ —known under the name of the blancmange function or the Takagi fractal curve—is nowhere differentiable. As such, it does not have a closed-form formula as any closed-form formula on a real interval must define a function and have a derivative at every point of that interval, except for a non-dense set of its points. Since $\tilde{F}$ can be expressed in a function, described by a closed-form formula, of the right-hand side of Formula (14), the latter does not have a closed-form formula either.

Theorem 3. For every closed-form formula $\varphi(n)$ there is a positive n such that

$$\sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}\left(\frac{n}{2^{k+1}}\right) \neq \varphi(n), \quad (29)$$

where Zigzag is a function defined by (8) and visualized in Figure 5.

Proof. Follows from the above discussion. A more detailed analytic proof is deferred to Section 10. $\square$

This way, we arrived at the following conclusion.

Corollary 3. There is no closed-form formula for $B(n)$.

Proof. A closed-form formula for $B(n)$ would, by virtue of (15) yield a closed-form formula for $\sum_{k=0}^{\lfloor \lg n \rfloor} 2^k \text{Zigzag}(\frac{n}{2^{k+1}})$, which by Theorem 3 does not exist. $\square$

Note 4. One can apply the reverse transformations to those used in Section 5 on the function $\tilde{F}$ and construct a fractal function $\check{F}$, shown in Figure 11, given by the equation

$$\check{F}(x) = 2^{\lfloor \lg x \rfloor} \tilde{F}\left(\frac{x}{2^{\lfloor \lg x \rfloor}}\right), \quad (30)$$

which for every positive integer n satisfies

$$\check{F}(n) = F(n), \quad (31)$$

where F is given by (19).

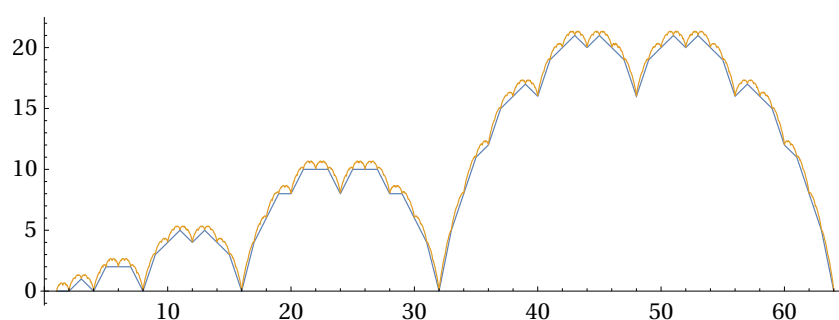


Figure 11. A graph (in orange) of function $\tilde{F}(x) = 2^{\lfloor \lg x \rfloor} \tilde{F}(\frac{x}{2^{\lfloor \lg x \rfloor}})$ plotted above a graph (in blue) of the function $F(x) = \sum_{k=1}^{\lfloor \lg x \rfloor} 2^k \text{Zigzag}(\frac{x}{2^k})$.

6. Computing $\tilde{F}(x)$ and $B(n)$ from One Another

Computing values of the function $\tilde{F}(x)$ does not have to be as complex as (or more complex than) definition (27) implies. Of course, for every integer n , $\tilde{F}(n) = 0$. One can apply some elementary arguments based on a structure visualized in Figure 10 to conclude that

$$\tilde{F}\left(\frac{2}{3}\right) = \tilde{F}\left(\frac{1}{3}\right) = \frac{2}{3} \quad (32)$$

(the latter being the maximum of $\tilde{F}(x)$) or that, for every positive integer k ,

$$\tilde{F}\left(\frac{1}{2^k}\right) = \frac{k}{2^k}. \quad (33)$$

It takes a bit more work to compute

$$\tilde{F}\left(\frac{3}{2^k}\right) = \frac{3k-4}{2^k}. \quad (34)$$

It turns out that computing values of the function $\tilde{F}(x)$ for every x that has a finite binary representation can be done easily if an oracle for computing the values of the function $B(n)$ defined by (2) and (4) is given, which is not that surprising after a glance in Figure 11. Once that is accomplished, since $\tilde{F}(x)$ is a continuous function and the set of numbers with finite binary representations is dense in the set $\mathfrak{R}$ of reals, it allows for fast approximations of $\tilde{F}(x)$ for every real x . (It helps to remember that $\tilde{F}$ is a periodic function with $\tilde{F}(x) = \tilde{F}(x - \lfloor x \rfloor)$.)

Theorem 4. For every positive integer n and integer k with $n \leq 2^k$,

$$\tilde{F}\left(\frac{n}{2^k}\right) = \frac{n \times k - 2B(n)}{2^k}. \quad (35)$$

Proof. Equality (35) can be verified experimentally, for instance, with the help of software for symbolic computation (Section 4.4). The analytic proof is deferred to Section 11. $\square$

Theorem 4 allows for easy computing of $B(n)$ if $\tilde{F}(\frac{n}{2^k})$ is given for some $k \geq \lg n$ using this form of (35):

Corollary 4. For every positive integer n and integer k with $n \leq 2^k$,

$$B(n) = \frac{n \times k}{2} - 2^{k-1} \tilde{F}\left(\frac{n}{2^k}\right). \quad (36)$$

Proof. An obvious conclusion from (35). $\square$

For instance, putting $k = \lceil \lg n \rceil + 1$ in (36) easily yields (15). For $k = \lceil \lg n \rceil$, I obtain

$$\begin{aligned} B(n) &= \frac{n \lceil \lg n \rceil}{2} - 2^{\lceil \lg n \rceil - 1} \tilde{F}\left(\frac{n}{2^{\lceil \lg n \rceil}}\right) = \\ &\text{[by (27)]} \\ &= \frac{n \lceil \lg n \rceil}{2} - 2^{\lceil \lg n \rceil - 1} \sum_{i=0}^{\infty} \frac{1}{2^i} \text{Zigzag}\left(2^i \frac{n}{2^{\lceil \lg n \rceil}}\right) = \\ &\text{[since for } i \geq \lceil \lg n \rceil, 2^i \frac{n}{2^{\lceil \lg n \rceil}} \text{ is integer and } \text{Zigzag}\left(2^i \frac{n}{2^{\lceil \lg n \rceil}}\right) = 0] \\ &= \frac{n \lceil \lg n \rceil}{2} - 2^{\lceil \lg n \rceil - 1} \sum_{i=0}^{\lceil \lg n \rceil - 1} \frac{1}{2^i} \text{Zigzag}\left(2^i \frac{n}{2^{\lceil \lg n \rceil}}\right) = \\ &= \frac{n \lceil \lg n \rceil}{2} - \frac{1}{2} \sum_{i=0}^{\lceil \lg n \rceil - 1} 2^{\lceil \lg n \rceil - i} \text{Zigzag}\left(\frac{n}{2^{\lceil \lg n \rceil - i}}\right). \end{aligned}$$

Substituting k for $\lceil \lg n \rceil - i$, I conclude

$$B(n) = \frac{n \lceil \lg n \rceil}{2} - \frac{1}{2} \sum_{k=1}^{\lceil \lg n \rceil} 2^k \text{Zigzag}\left(\frac{n}{2^k}\right), \quad (37)$$

similar to the (15) characterization of $B(n)$.

7. Relationship Between the Best Case and the Worst Case

An attentive student of MergeSort tends to believe (taking into account that substituting $n - 1$ for $\lfloor \frac{n}{2} \rfloor$ in the recurrence relation (3) for the best case yields the recurrence relation for the worst case) that its worst-case behavior is about twice as bad as its best-case behavior. This, of course, is only approximately true. In this section, I will derive the exact difference between $2B(n)$ and $W(n)$ using the function F defined by (17).

An exact formula for the number $W(n)$ of comparisons of keys performed by MergeSort in the worst case is known (see Formula (A7) in Appendix A) and is given for any positive integer n by the following equality:

$$W(n) = \sum_{i=1}^n \lceil \lg i \rceil. \quad (38)$$

From (15) and (17), one can derive

$$\begin{aligned}
 2B(n) &= n \lfloor \lg n \rfloor - 2^{\lfloor \lg n \rfloor + 1} + 2n - F(n) = \\
 &[\text{by } \sum_{i=1}^n \lceil \lg i \rceil = n \lfloor \lg n \rfloor - 2^{\lfloor \lg n \rfloor + 1} + n + 1 \text{ from [3]}] \\
 &= \sum_{i=1}^n \lceil \lg i \rceil - 1 + n - F(n) = \\
 &[\text{by (38)}] \\
 &= W(n) - 1 + n - F(n).
 \end{aligned}$$

The above equality yields the following characterization.

Theorem 5. *For every positive integer n , the difference between twice the number $B(n)$ of a comparison of keys performed in the best case and the number $W(n)$ of a comparison of keys performed in the worst case by MergeSort while sorting an n -element array is*

$$2B(n) - W(n) = n - 1 - F(n), \quad (39)$$

where $F(n)$, visualized in Figure 8, is given by (19).

Proof. Follows from the above discussion. $\square$

In particular, since, for every positive integer n ,

$$0 \leq F(n) \leq \frac{n-1}{2} \quad (40)$$

(see Figure 8 for an explanation), I conclude with the following tight linear bounds on $2B(n) - W(n)$.

Corollary 5. *For every positive integer n , the difference between twice the minimum number $B(n)$ and the maximum number $W(n)$ of a comparison of keys performed in the worst case by MergeSort while sorting an n -element array satisfies this inequality:*

$$\frac{n-1}{2} \leq 2B(n) - W(n) \leq n - 1. \quad (41)$$

Proof. Follows from (39) and (40). $\square$

Obviously, $2B(n) - W(n) = n - 1$ whenever $F(n) = 0$, that is, whenever $n = 2^{\lfloor \lg n \rfloor}$. It can be shown that $2B(n) - W(n) = \frac{n-1}{2}$ whenever $n = \frac{1}{3}(2^{k+1} + (-1)^k)$ for some integer $k \geq 0$. Therefore, the number of comps performed by MergeSort in the best case is larger than half the number of comps performed by it in the worst case by at least by $\frac{n-1}{4}$ and at most by $\frac{n-1}{2}$, which makes the former number significantly larger (by a term of size $\Theta(n)$) than half of the latter number.

A graph of $2B(n) - W(n)$ and its tight bounds are shown in Figure 12.

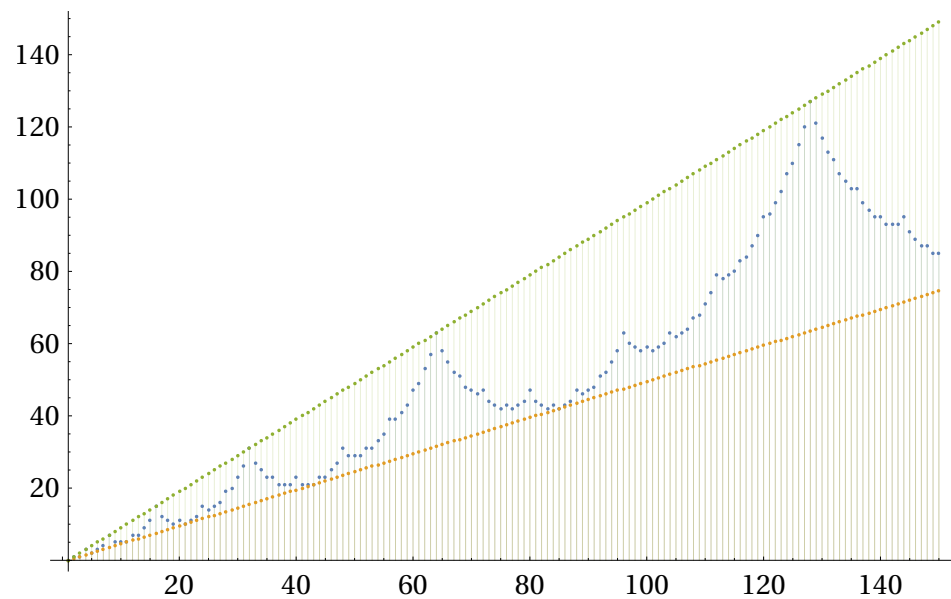


Figure 12. A graph of $2B(n) - W(n)$ shown (in blue) between graphs of its tight linear bounds $n - 1$ (green) and $\frac{n-1}{2}$ (in orange).

8. The Sum of Digits Problem

A known [but not to me during my derivation of Formula (15)] explicit formula, published in [4], for the total number $A(n, 2)$ of 1s in binary representations of all integers between 0 and n (not including 0 and n) is expressed in terms of the function *Zigzag* (referred to as $2g$ in [4]). The following are verbatim quotations and screenshots from [4].

“If $\alpha(\kappa, r)$ denotes the sum of the digits of κ when κ is represented in base r , then”

$$(1.1) \quad A(n, r) \equiv \sum_{\kappa < n} \alpha(\kappa, r) = \frac{1}{2}(r - 1)n \log_r n - E(n, r)$$

“Let $g(x)$ be periodic of period 1 and defined on $[0, 1]$ by”

$$(1.2) \quad g(x) = \begin{cases} \frac{1}{2}x, & 0 \leq x \leq \frac{1}{2} \\ \frac{1}{2}(1 - x), & \frac{1}{2} < x \leq 1. \end{cases}$$

The function

$$(1.3) \quad f(x) \equiv \sum_{i=0}^{\infty} \frac{1}{2^i} g(2^i x)$$

can be shown to be nondifferentiable. The relation between this function and $E(n, 2)$ is demonstrated in the following theorems:

THEOREM 1. *If the integer n is written $n = 2^m(1 + x)$, $0 \leq x < 1$, then*

$$E(n, 2) = 2^{m-1} \{ 2f(x) + (1 + x) \log_2 (1 + x) - 2x \}.$$

It has been shown in [5] that the recurrence relation for $A(n, 2)$ is the same as the recurrence relation for $B(n)$ given by (2) and (3). Therefore, Formula (14) derived in this article is equivalent to $A(n, 2)$ given above by the considerably more complicated definition. Interestingly, the above definition can be simplified to (14) along the lines of the elementary derivation of the alternative Formula (37) for $B(n)$.

Even more interestingly, if someone did succeed in simplifying Trollope’s formula in [4] then I am not aware of it. In particular, the most recent article [6] on the subject of sum of digits does not contain any hint that such a simplification has been found and published.

9. Proof of Theorem 1, Section 4.4

In this section, I provide an analytic proof of the experimentally derived Theorem 1, Section 4.4, which was instrumental for the derivation of a logarithmic-length Formula (15) for $B(n)$. The result and its proof have a flavor of Concrete Mathematics. Although they are interesting in their own right, they cannot be found in [7].

Theorem 6. (Same as Theorem 1.) *For every natural number n and every positive natural number m ,*

$$\sum_{i=m}^{2m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor - \sum_{i=0}^{m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor = 2m \times \text{Zigzag} \left(\frac{n}{2m} \right), \quad (42)$$

where Zigzag is a function defined by (8) and visualized in Figure 5.

Proof. First, let us note that

$$\begin{aligned} \sum_{i=m}^{2m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor - \sum_{i=0}^{m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor &= \sum_{i=0}^{m-1} \left\lfloor \frac{n+i+m}{2m} \right\rfloor - \sum_{i=0}^{m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor = \\ &= \sum_{i=0}^{m-1} \left(\left\lfloor \frac{n+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{n+i}{2m} \right\rfloor \right), \end{aligned}$$

that is,

$$\sum_{i=m}^{2m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor - \sum_{i=0}^{m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor = \sum_{i=0}^{m-1} \left(\left\lfloor \frac{n+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{n+i}{2m} \right\rfloor \right). \quad (43)$$

Let

$$n = k \times 2m + r, \quad (44)$$

where $0 \leq r < 2m$, and let $0 \leq i < m$. We have

$$\left\lfloor \frac{n+i}{2m} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{k \times 2m + r + i}{2m} + \frac{1}{2} \right\rfloor = k + \left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor$$

and

$$\left\lfloor \frac{n+i}{2m} \right\rfloor = \left\lfloor \frac{k \times 2m + r + i}{2m} \right\rfloor = k + \left\lfloor \frac{r+i}{2m} \right\rfloor.$$

Thus, by virtue of (43),

$$\sum_{i=m}^{2m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor - \sum_{i=0}^{m-1} \left\lfloor \frac{n+i}{2m} \right\rfloor = \sum_{i=0}^{m-1} \left(\left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{r+i}{2m} \right\rfloor \right). \quad (45)$$

We have

$$\left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{r+i}{2m} \right\rfloor = \begin{cases} 1 & \text{if } \frac{1}{2} \leq \frac{r+i}{2m} < 1 \\ 0 & \text{otherwise,} \end{cases} \quad (46)$$

because $\frac{r+i}{2m} + \frac{1}{2} < \frac{3m}{2m} + \frac{1}{2} = 2$ so that $\left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor \leq 1$, and therefore, $\left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{r+i}{2m} \right\rfloor \leq 1$.

Let I be defined as

$$I = \{i \in \mathbb{N} \mid \frac{1}{2} \leq \frac{r+i}{2m} < 1\} = \{i \in \mathbb{N} \mid m-r \leq i < 2m-r\}. \quad (47)$$

By virtue of (46), we have

$$\sum_{i=0}^{m-1} \left(\left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{r+i}{2m} \right\rfloor \right) = \sum_{i \in I} \left(\left\lfloor \frac{r+i}{2m} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{r+i}{2m} \right\rfloor \right) = \sum_{i \in I} 1 = \#I, \quad (48)$$

where $\#(I)$ denotes the cardinality of I .

If $r \leq m$, then, by (47), $\#I = m - (m - r) = r$. If $r > m$, then, by (47), $\#I = 2m - r$. In any case,

$$\#I = \min(r, 2m - r) = 2m \min\left(\frac{r}{2m}, 1 - \frac{r}{2m}\right) =$$

[since $0 \leq \frac{r}{2m} < 1$ so that $\lfloor \frac{r+i}{2m} \rfloor = 0$ and $\lceil \frac{r+i}{2m} \rceil = 1$]

$$= 2m \min\left(\frac{r}{2m} - \lfloor \frac{r}{2m} \rfloor, \lceil \frac{r}{2m} \rceil - \frac{r}{2m}\right) =$$

[by definition (8) of the function *Zigzag*]

$$= 2m \times \text{Zigzag}\left(\frac{r}{2m}\right) =$$

[since *Zigzag* is a periodic function with period 1]

$$= 2m \times \text{Zigzag}\left(k + \frac{r}{2m}\right) = 2m \times \text{Zigzag}\left(\frac{k \times 2m + r}{2m}\right) =$$

[by (44)]

$$= 2m \times \text{Zigzag}\left(\frac{n}{2m}\right).$$

Thus

$$\#I = 2m \times \text{Zigzag}\left(\frac{n}{2m}\right). \quad (49)$$

From (43), (45), (48), and (49), I conclude (42). $\square$

10. Proof of Theorem 3, Section 5

In this section, I provide an analytic proof of Theorem 3.

I begin with a brief discussion/motivation of what can be generally considered a *closed-form formula* for a function from the set of real numbers, except, perhaps, a finite number of reals, into a set of real numbers.

The general concept of *closed-form formula* (*cff*) has not been precisely defined in the literature. In this article, it denotes a formula that is composed of constants, variables, and some “standard” arithmetic and logical operations that has a finite and fixed structure and length that do not depend on the values of any variables occurring in the said formula. More specifically, in this article, it is used in the following sense.

The arithmetic constants are any integers with fixed values. The logical constants are *true* and *false*.

The basic arithmetic functions are: binary addition, binary multiplication, binary maximum and minimum, subtraction, division, exponentiation, logarithm functions, and the floor function. The basic arithmetic relations are: the equality relation and the less-than relation. The basic logical operations are: negation and binary conjunction.

An atomic *cff* is an expression that is a constant or variable, or denotes a reference to a basic function or relation whose arguments are constants or variables. The set of *cffs* is the smallest set of expressions that contains all atomic formulas and is closed under any finite and syntactically correct applications of basic functions and relations and the conditional arithmetic operation by the statement *if c then d else e*, where *c* is a logical *cff* and *d* and *e* are arithmetic *cffs*.

Note 5. One could use a more inclusive definition of *closed-form formula* without invalidating the results presented in this article as long as the extra arithmetic expressions allowed by it are differentiable on their domains except, perhaps, on their non-dense subsets.

Since non-rational real numbers are defined as limits of infinite, Cauchy-convergent rationals, limits of certain infinite sequences of reals do implicitly occur in definitions of some functions the references to which are accepted as *cffs*. Below is a motivating example of the function $2^x : \mathbb{R} \rightarrow \mathbb{R}$ as an insight of what is accepted as a *cff* for a *continuous* function—like, say, $\tilde{F}(x)$ —on the set $\mathbb{R}$ of reals or on an interval thereof. One picks a dense (in the metric topology of $\mathbb{R}$) subset $\mathbb{Q}$ (in this example, the set of rationals) of $\mathbb{R}$, with a collection of mappings $\rho_x(i) : \mathbb{N} \rightarrow \mathbb{Q}$, where $x \in \mathbb{R}$, given by $\rho_x(i) = \frac{[i \times x]}{i}$ so that $\lim_{i \rightarrow \infty} \rho_x(i) = x$. Since, for any $x \in \mathbb{R} \setminus \mathbb{Q}$, 2^x has been defined as

$$2^x = \lim_{i \rightarrow \infty} 2^{\rho_x(i)} = \lim_{i \rightarrow \infty} \sqrt[i]{2^{[i \times x]}}, \quad (50)$$

$\lim_{i \rightarrow \infty} \sqrt[i]{2^{[i \times x]}}$ is considered a *cff* for the function $2^x : \mathbb{R} \rightarrow \mathbb{R}$.

For the reader's convenience, Theorem 3 is quoted below as Theorem 7.

Theorem 7. (Same as Theorem 3). *There is no cff $\varphi(n)$ the domain of which contains all positive integers and the values of which coincide with $\sum_{k=0}^{[\lg n]} 2^k \text{Zigzag}(\frac{n}{2^{k+1}})$ for all positive integers n ; that is, for every cff $\varphi(n)$, there is a positive n such that*

$$\sum_{k=0}^{[\lg n]} 2^k \text{Zigzag}(\frac{n}{2^{k+1}}) \neq \varphi(n), \quad (51)$$

where *Zigzag* is a function defined by (8) and visualized in Figure 5.

The rest of this section constitutes the proof of Theorem 7.

Lemma 1. *For every positive integer n ,*

$$\tilde{F}(\frac{n}{2^{[\lg n]}}) = \frac{n([\lg n] + 2) - 2B(n)}{2^{[\lg n]}} - 2, \quad (52)$$

where the function F has been defined by equality (17); the function $\tilde{F}$, visualized in Figure 13 has been defined by equality (27); and the function $B(n)$ has been defined by Equations (2) and (4).

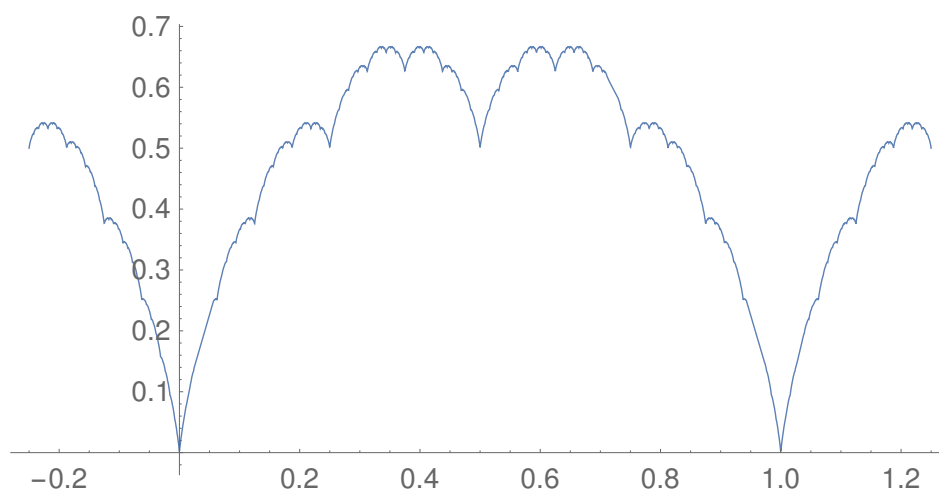


Figure 13. A graph of the function $\tilde{F}(x) = \sum_{i=0}^{\infty} \frac{1}{2^i} \text{Zigzag}(2^i x)$.

Proof. From (17) I compute

$$\sum_{i=0}^{\lfloor \lg n \rfloor} 2^{i+1} \text{Zigzag}\left(\frac{n}{2^{i+1}}\right) = F(n) + 2^{\lfloor \lg n \rfloor + 1} - n,$$

that is,

$$\sum_{i=0}^{\lfloor \lg n \rfloor} 2^i \text{Zigzag}\left(\frac{n}{2^{i+1}}\right) = \frac{1}{2}F(n) + 2^{\lfloor \lg n \rfloor} - \frac{n}{2}. \quad (53)$$

Applying equality (53) to equality (15) I conclude

$$B(n) = \frac{n}{2}(\lfloor \lg n \rfloor + 1) - \left(\frac{1}{2}F(n) + 2^{\lfloor \lg n \rfloor} - \frac{n}{2}\right),$$

or

$$B(n) = \frac{n}{2}(\lfloor \lg n \rfloor + 1) - \frac{1}{2}F(n) - 2^{\lfloor \lg n \rfloor} + \frac{n}{2},$$

that is,

$$\frac{1}{2}F(n) = \frac{n}{2}(\lfloor \lg n \rfloor + 1) - B(n) - 2^{\lfloor \lg n \rfloor} + \frac{n}{2},$$

or

$$F(n) = n(\lfloor \lg n \rfloor + 2) - 2B(n) - 2^{\lfloor \lg n \rfloor + 1}. \quad (54)$$

On the other hand, by virtue of (19),

$$F(n) = \sum_{i=1}^{\lfloor \lg n \rfloor} 2^i \text{Zigzag}\left(\frac{n}{2^i}\right) =$$

[putting $j = \lfloor \lg n \rfloor - i$]

$$= \sum_{j=0}^{\lfloor \lg n \rfloor - 1} 2^{\lfloor \lg n \rfloor - j} \text{Zigzag}\left(\frac{n}{2^{\lfloor \lg n \rfloor - j}}\right) = 2^{\lfloor \lg n \rfloor} \sum_{j=0}^{\lfloor \lg n \rfloor - 1} \frac{1}{2^j} \text{Zigzag}\left(\frac{2^j n}{2^{\lfloor \lg n \rfloor}}\right) =$$

[since for $j \geq \lfloor \lg n \rfloor$, $\frac{2^j n}{2^{\lfloor \lg n \rfloor}} \in \mathbb{N}$ so that $\text{Zigzag}\left(\frac{2^j n}{2^{\lfloor \lg n \rfloor}}\right) = 0$]

$$= 2^{\lfloor \lg n \rfloor} \sum_{j=0}^{\infty} \frac{1}{2^j} \text{Zigzag}\left(\frac{2^j n}{2^{\lfloor \lg n \rfloor}}\right) =$$

[by (27)]

$$= 2^{\lfloor \lg n \rfloor} \tilde{F}\left(\frac{n}{2^{\lfloor \lg n \rfloor}}\right).$$

Thus

$$F(n) = 2^{\lfloor \lg n \rfloor} \tilde{F}\left(\frac{n}{2^{\lfloor \lg n \rfloor}}\right) \quad (55)$$

or

$$\tilde{F}\left(\frac{n}{2^{\lfloor \lg n \rfloor}}\right) = \frac{1}{2^{\lfloor \lg n \rfloor}} F(n). \quad (56)$$

Combining equalities (54) and (56) yields

$$\tilde{F}\left(\frac{n}{2^{\lfloor \lg n \rfloor}}\right) = \frac{1}{2^{\lfloor \lg n \rfloor}} (n(\lfloor \lg n \rfloor + 2) - 2B(n) - 2^{\lfloor \lg n \rfloor + 1}),$$

or (52). $\square$

Lemma 2. If the function $B(n)$ defined by equations (2) and (4) has a cff $\beta : \mathbb{N} \rightarrow \mathbb{N}$, then the function $\tilde{F}(x)$ defined by equation (27) has a cff $\varphi : [1, 2) \rightarrow [0, \frac{2}{3}]$.

Proof. Let

$$D = \left\{ \frac{n}{2^{\lfloor \lg n \rfloor}} \mid n \in \mathbb{N} \right\} \quad (57)$$

be the set of rationals in the interval $[1, 2)$ with finite binary representations, enumerated by $v(n) : \mathbb{N} \rightarrow D$ given by $v(n) = \frac{n}{2^{\lfloor \lg n \rfloor}}$ and visualized in Figure 14. (It is a trivial exercise to show that every real number with finite binary representation in the interval $[1, 2)$ is of the form $\frac{n}{2^{\lfloor \lg n \rfloor}}$ for some $n \in \mathbb{N}$, and it is obvious that every real number of that form has a finite binary representation and falls into that interval.)

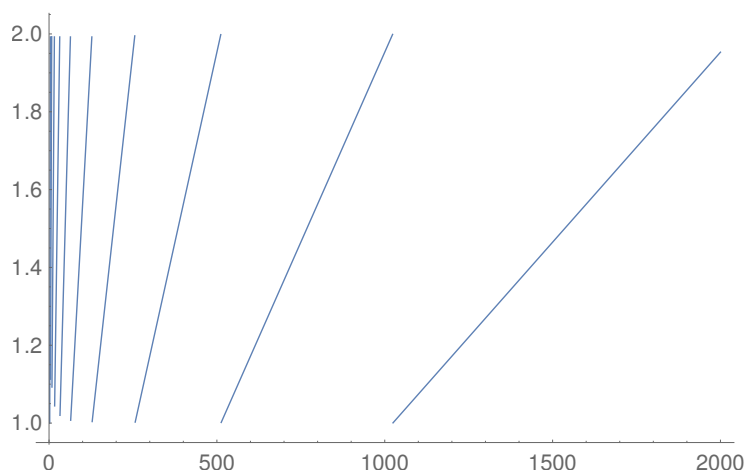


Figure 14. A graph of enumeration $v(n) = \frac{n}{2^{\lfloor \lg n \rfloor}}$ of the set D .

D is a dense subset of the interval $[1, 2)$ of reals. Indeed, if $x \in [1, 2)$, then, for every $n \in \mathbb{N}$, $\frac{\lfloor 2^n x \rfloor}{2^n} \in D$ and

$$\lim_{n \rightarrow \infty} \frac{\lfloor 2^n x \rfloor}{2^n} = x. \quad (58)$$

Hence, for any $x \in [1, 2)$, putting

$$n = \lfloor 2^i x \rfloor, \quad (59)$$

so that

$$\lfloor \lg n \rfloor = \lfloor \lg \lfloor 2^i x \rfloor \rfloor = \lfloor \lg 2^i x \rfloor = \lfloor i + \lg x \rfloor = i + \lfloor \lg x \rfloor = i$$

[the last equality holds because $1 \leq x < 2$ so that $0 \leq \lg x < 1$ and $\lfloor \lg x \rfloor = 0$], or

$$\lfloor \lg n \rfloor = i, \quad (60)$$

I conclude, by virtue of (58),

$$\tilde{F}(x) = \tilde{F}\left(\lim_{i \rightarrow \infty} \frac{\lfloor 2^i x \rfloor}{2^i}\right) =$$

[by the continuity of $\tilde{F}(x)$]

$$= \lim_{i \rightarrow \infty} \tilde{F}\left(\frac{\lfloor 2^i x \rfloor}{2^i}\right) =$$

[by equality (52) of Lemma 1]

$$= \lim_{i \rightarrow \infty} \frac{\lfloor 2^i x \rfloor (i + 2) - 2B(\lfloor 2^i x \rfloor)}{2^i} - 2 = \lim_{i \rightarrow \infty} \frac{i \lfloor 2^i x \rfloor - 2B(\lfloor 2^i x \rfloor)}{2^i} + \lim_{i \rightarrow \infty} 2 \frac{\lfloor 2^i x \rfloor}{2^i} - 2 =$$

[by equality (58)]

$$\lim_{i \rightarrow \infty} \frac{i \lfloor 2^i x \rfloor - 2B(\lfloor 2^i x \rfloor)}{2^i} + 2x - 2.$$

Thus for any $x \in [1, 2)$,

$$\tilde{F}(x) = \lim_{i \rightarrow \infty} \frac{i \lfloor 2^i x \rfloor - 2B(\lfloor 2^i x \rfloor)}{2^i} + 2x - 2. \quad (61)$$

Equality (61) shows that if there is a *cff* $\beta : \mathbb{N} \rightarrow \mathbb{N}$ for the function B defined by Equations (2) and (4) then there is a *cff* $\varphi : [1, 2) \rightarrow [0, \frac{2}{3}]$ given by

$$\lim_{i \rightarrow \infty} \frac{i \lfloor 2^i x \rfloor - 2\beta(\lfloor 2^i x \rfloor)}{2^i} + 2x - 2$$

for $\tilde{F}(x)$. This completes the proof of Lemma 2. $\square$

Should the nowhere differentiable function $\tilde{F}$ have a *cff*, it would be differentiable everywhere except, perhaps, on a non-dense subset of $\mathbb{R}$. The following inductive argument demonstrates that. All atomic *cffs* are differentiable except, perhaps, on a non-dense subset of $\mathbb{R}$. If a finite number of *cffs* are differentiable except, perhaps, on non-dense subsets of $\mathbb{R}$, then their composition is differentiable except, perhaps, on non-dense subsets of $\mathbb{R}$. (For instance, the function $\sqrt{x} \text{Zigzag}(\frac{1}{x})$ is differentiable on $(0, 1)$, except for the non-dense set $\{\frac{2}{n} \mid n \in \mathbb{N}\}$.) Thus $\tilde{F}$ has no *cff*.

The above observation, together with Lemma 2, completes the proof of Theorem 7.

11. Proof of Theorem 4, Section 6

In this Section, I provide an analytic proof of experimentally derived Theorem 4, Section 6. This result, re-stated as Theorem 8 below, allows for practically efficient computations of values of the continuous blancmange function for reals with finite binary floating-point representations. I also provide some properties (Lemmas 3–5) of the *Zigzag* function, given by equality (8) and visualized in Figure 5, that are useful for a neat derivation of a formula for the blancmange function as the limit of a finite sum of some values of the *Zigzag* function.

Let the function $\tilde{F}$ (known as the blancmange function), visualized in Figure 13, be defined by (27) and $B(n)$, given by (2) and (4), be the least number of comparisons of keys that MergeSort performs while sorting an n -element array.

Theorem 8. (Same as Theorem 4.) *For every positive integer n and integer k with $n \leq 2^k$,*

$$\tilde{F}\left(\frac{n}{2^k}\right) = \frac{n \times k - 2B(n)}{2^k}. \quad (62)$$

The remainder of this section constitutes a proof of Theorem 8.

Note. The function *Zigzag*, visualized in Figure 5, has been defined by Equation (8).

Lemma 3. *For every $k \geq \lfloor \lg n \rfloor + 2$,*

$$2^k \text{Zigzag}\left(\frac{n}{2^k}\right) = n. \quad (63)$$

Proof. Let $k \geq \lfloor \lg n \rfloor + 2$, or $2^k \geq 2 \times 2^{\lfloor \lg n \rfloor + 1} > 2n$, that is,

$$2^k > 2n. \quad (64)$$

We have

$$0 \leq \left\lfloor \frac{n}{2^k} \right\rfloor \leq$$

[by (64)]

$$\leq \lfloor \frac{n}{2^n} \rfloor = \lfloor \frac{1}{2} \rfloor = 0$$

or

$$\lfloor \frac{n}{2^k} \rfloor = 0. \quad (65)$$

Additionally,

$$1 \leq \lceil \frac{n}{2^k} \rceil \leq$$

[by (64)]

$$\leq \lceil \frac{n}{2^n} \rceil = \lceil \frac{1}{2} \rceil = 1$$

or

$$\lceil \frac{n}{2^k} \rceil = 1. \quad (66)$$

Now,

$$2^k \text{Zigzag}(\frac{n}{2^k}) =$$

[by (8)]

$$= 2^k \min\{\frac{n}{2^k} - \lfloor \frac{n}{2^k} \rfloor, \lceil \frac{n}{2^k} \rceil - \frac{n}{2^k}\} = \min\{n - 2^k \lfloor \frac{n}{2^k} \rfloor, 2^k \lceil \frac{n}{2^k} \rceil - n\} =$$

[by (65) and (66)]

$$= \min\{n, 2^k - n\} =$$

[since by (64), $2^k - n \geq n$]

$$= n.$$

Hence, (63) holds. $\square$

Lemma 4. For every $k \geq \lfloor \lg n \rfloor + 1$,

$$\sum_{i=\lfloor \lg n \rfloor + 2}^k 2^i \text{Zigzag}(\frac{n}{2^i}) = n \times k - n(\lfloor \lg n \rfloor + 1). \quad (67)$$

Proof. By induction on k .

Basis step: $k = \lfloor \lg n \rfloor + 1$.

$$L = \sum_{i=\lfloor \lg n \rfloor + 2}^{\lfloor \lg n \rfloor + 1} 2^i \text{Zigzag}(\frac{n}{2^i}) = 0.$$

$$R = n(\lfloor \lg n \rfloor + 1) - n(\lfloor \lg n \rfloor + 1) = 0.$$

Hence, $L = R$. This completes the *basis step*.

Inductive step: $k \geq \lfloor \lg n \rfloor + 2$.

Inductive hypothesis: (67).

$$\sum_{i=\lfloor \lg n \rfloor + 2}^{k+1} 2^i \text{Zigzag}(\frac{n}{2^i}) = \sum_{i=\lfloor \lg n \rfloor + 2}^k 2^i \text{Zigzag}(\frac{n}{2^i}) + 2^k \text{Zigzag}(\frac{n}{2^k}) =$$

[by the *inductive hypothesis* and by equality (63) of Lemma 3]

$$= n \times k - n(\lfloor \lg n \rfloor + 1) + n = n \times (k + 1) - n(\lfloor \lg n \rfloor + 1).$$

Thus

$$\sum_{i=\lfloor \lg n \rfloor + 2}^{k+1} 2^i \text{Zigzag}\left(\frac{n}{2^i}\right) = n \times (k+1) - n(\lfloor \lg n \rfloor + 1).$$

This completes the *inductive step*. $\square$

Lemma 5. For every $k \geq \lfloor \lg n \rfloor + 1$,

$$2^k \tilde{F}\left(\frac{n}{2^k}\right) = \sum_{i=1}^k 2^i \text{Zigzag}\left(\frac{n}{2^i}\right). \quad (68)$$

Proof. By definition (27) of the function $\tilde{F}$, I get

$$2^k \tilde{F}\left(\frac{n}{2^k}\right) = 2^k \sum_{i=0}^{\infty} \frac{1}{2^i} \text{Zigzag}\left(2^i \frac{n}{2^k}\right) =$$

[since, for every integer x , $\text{Zigzag}(x) = 0$, so that, for $i \geq k$, $\text{Zigzag}(2^i \frac{n}{2^k}) = 0$]

$$= 2^k \sum_{i=0}^{k-1} \frac{1}{2^i} \text{Zigzag}\left(2^i \frac{n}{2^k}\right) = \sum_{i=0}^{k-1} 2^{k-i} \text{Zigzag}\left(\frac{n}{2^{k-i}}\right) =$$

[putting $j = k - i$]

$$= \sum_{j=1}^k 2^j \text{Zigzag}\left(\frac{n}{2^j}\right),$$

which completes the proof of (68). $\square$

At this point, I am ready to conclude the proof of Theorem 4.

By virtue of (15), we have

$$\begin{aligned} 2B(n) &= n(\lfloor \lg n \rfloor + 1) - \sum_{k=0}^{\lfloor \lg n \rfloor} 2^{k+1} \text{Zigzag}\left(\frac{n}{2^{k+1}}\right) = \\ &= n(\lfloor \lg n \rfloor + 1) - \sum_{i=1}^{\lfloor \lg n \rfloor + 1} 2^i \text{Zigzag}\left(\frac{n}{2^i}\right) = \\ &= n(\lfloor \lg n \rfloor + 1) - \sum_{i=1}^k 2^i \text{Zigzag}\left(\frac{n}{2^i}\right) + \sum_{i=\lfloor \lg n \rfloor + 2}^k 2^i \text{Zigzag}\left(\frac{n}{2^i}\right) = \end{aligned}$$

[by Lemmas 4 and 5]

$$= n(\lfloor \lg n \rfloor + 1) - 2^k \tilde{F}\left(\frac{n}{2^k}\right) + n \times k - n(\lfloor \lg n \rfloor + 1) = n \times k - 2^k \tilde{F}\left(\frac{n}{2^k}\right),$$

that is,

$$2B(n) = n \times k - 2^k \tilde{F}\left(\frac{n}{2^k}\right),$$

from which (63) follows.

This completes the proof of Theorem 4.

Note. A glance at the proof of Lemma 4 suffices to notice that it fails if $n > 2^k$, and so does Theorem 4. In particular, for $k = \lfloor \lg n \rfloor$, Lemma 1 yields

$$\tilde{F}\left(\frac{n}{2^k}\right) = \frac{n \times k - 2B(n)}{2^k} + \frac{2n}{2^k} - 2 > \frac{n \times k - 2B(n)}{2^k} \quad (69)$$

since, for $n > 2^{\lceil \lg n \rceil}$, $\frac{2n}{2^{\lceil \lg n \rceil}} - 2 > 0$.

12. Final Remarks

It appears that a lack of sufficient interaction between research related to the sum of digits problem and analysis of recursive algorithms had a detrimental impact on the progress of the said research. In particular, I found that some clever use of recursion trees that are often utilized in the analysis of algorithms but are relatively rarely used in pure mathematics allowed for a relatively easy derivation of a finitary Formula (7) that, due to its finiteness, is significantly simpler and more practical to compute (but still requiring the addition of $n - 1$ terms) than an instance for $r = 2$ of the infinite formula (1.1) derived in [4] and quoted in Section 8. I suppose that if the mathematicians interested in the sum of digits problem knew Formula (7), then some of them would simplify it to Formula (15) of the [Main Theorem](#) that can be evaluated in $O(\log^c n)$ for some constant c .

A similar lack of communication between researchers has been hypothesized in [6]: “When reading the literature on the subject we have noted that the most recent papers do not always cite more ancient ones, confirming a remark of Stolarsky [...]: *Whatever its mathematical virtues, the literature on sums of digital sums reflects a lack of communication between researchers*”.

On the other hand, those willing to find or quote from the literature a formula for the exact number of comparisons of keys performed by MergeSort in the best case on an n -element array might have been discouraged by finding Trollope’s infinite formula (1.1) that, for $r = 2$, yields the count of 1s occurring in binary representations of all integers between 1 and n and has been proved, around 1974, to yield the said number of comps. To make things look even more discouraging, directly counting the said 1s requires $\Theta(n \lg n)$ time that is about as long as running MergeSort on an increasingly sorted array of consecutive numbers from 1 to n and counting how many comps have been done. These could have been factors that stopped some of those analyzing MergeSort from trying to precisely characterize its best-case time complexity, thus leaving a significant loophole in many computer science faculty and students’ knowledge of the time complexity of MergeSort.

Fortunately, this article fills this important loophole.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The Author declares that no conflict of interest or ethical issues have occurred during the research described and in the preparation of this article.

Appendix A. A Derivation of the Worst-Case Running Time

$W(n) = \sum_{i=1}^n \lceil \lg i \rceil$ of MergeSort

Let us assume that $n \geq 2$ is large enough to spur a cascade of many recursive calls to MergeSort following the *recursion tree* T , a sketch of which is shown in Figure 2.

The nodes in tree T correspond to calls to MergeSort and show sizes of (sub)arrays passed to those calls. The root corresponds to the original call to MergeSort. If a call that is represented by a node p executes further recursive calls to MergeSort, then these calls are represented by the children of p ; otherwise, p is a leaf. Thus T is a 2-tree (a binary tree whose every non-leaf has exactly 2 children).

The levels in tree T are enumerated from 0 to h , where h is the number of the last level of the tree, or—in other words—the depth of T . In Figure 2, they are shown on the left side of the tree. The root is at the level 0, its children are at level 1, its grandchildren are at level 2, its great-grandchildren (not shown on the sketch in Figure 2) are at level 3, and so

on. Clearly, since every call to MergeSort on a sub-array of size ≥ 2 executes two further recursive calls to MergeSort, only the nodes that show the value 1 are leaves and all other nodes have 2 children each. Thus, since all nodes in the last level h are leaves, they all show the value 1. Additionally, since the original input array gets split, eventually, onto n 1-element sub-arrays, the number of all leaves in T is n . (This, however, does not mean that the last level h necessarily contains all the leaves of T).

If a level i has 2^i nodes, each of them showing a value ≥ 2 , then each such node has 2 children so that level $i + 1$ has twice the number of nodes in level i , that is, 2^{i+1} nodes. Since level 0 has 2^0 nodes, it follows (completion of a proof by induction with the basis and inductive steps outlined above is left as an exercise for the reader) that if k is the level number of any level above which all the nodes show values ≥ 2 , then all levels $i = 0, \dots, k$ contain exactly 2^i nodes each.

The last level h may contain 2^h nodes or less. I am going to show that each level i above level h contains exactly 2^i nodes. Here is a very insightful property that I am going to use for that purpose. It states that MergeSort is splitting its input array fairly evenly so that, at any level of the recursive tree, the difference between the lengths of the longest sub-array and the shortest sub-array is ≤ 1 . This fact is the root cause of good worst-case performance of MergeSort.

Theorem A1. *The difference between values shown by any two nodes in the same level of the recursion tree for MergeSort is ≤ 1 .*

Proof. The property clearly holds for level 0. I will show that if it holds for level i and i is not the last level of the recursion tree (that is, $i < h$), then it also holds for level $i + 1$.

Let us assume that the property holds for some level $i < h$. Let $c \leq d$ be numbers shown by any two (not necessarily distinct) nodes in level $i + 1$. It suffices to show that

$$d - c \leq 1. \quad (\text{A1})$$

Let $a \leq b$ be the numbers shown by the parents of the above-mentioned nodes. Those parents, of course, must reside in level i . By the inductive hypothesis (that holds for level i), $b - a \leq 1$, that is,

$$a \leq b \leq a + 1. \quad (\text{A2})$$

The numbers shown by all their four children are $\lfloor \frac{a}{2} \rfloor$, $\lceil \frac{a}{2} \rceil$, $\lfloor \frac{b}{2} \rfloor$, and $\lceil \frac{b}{2} \rceil$, so the largest difference between any of those four numbers is $\lceil \frac{b}{2} \rceil - \lfloor \frac{a}{2} \rfloor$. In particular, $d - c$ is not larger than that.

We have

$$d - c \leq \lceil \frac{b}{2} \rceil - \lfloor \frac{a}{2} \rfloor \leq$$

[by (A2)]

$$\leq \lceil \frac{a+1}{2} \rceil - \lfloor \frac{a}{2} \rfloor =$$

[since for any integer c , $\lceil \frac{c}{2} \rceil = \lfloor \frac{c+1}{2} \rfloor$]

$$= \lfloor \frac{a+2}{2} \rfloor - \lfloor \frac{a}{2} \rfloor = \lfloor \frac{a}{2} + 1 \rfloor - \lfloor \frac{a}{2} \rfloor =$$

[since for every x , $\lfloor x + 1 \rfloor = \lfloor x \rfloor + 1$]

$$= \lfloor \frac{a}{2} \rfloor + 1 - \lfloor \frac{a}{2} \rfloor = 1.$$

Thus (A1) holds. This completes the inductive step and completes the proof of the property. $\square$

As I have noted, the values shown at all nodes in the last level h are all 1. Thus the values shown at their parents that reside at level $h - 1$ are all 2, and the values shown at their grand parents, which reside at level $h - 2$, are all ≥ 3 . Thus, by Theorem A1, all nodes at level $h - 2$ show values ≥ 2 and, therefore (as I have proved before), all levels $i = 0, \dots, h - 1$ have 2^i nodes each, as it has been visualized in Figure 2.

Theorem A2. The depth h of the recursion tree $T(n)$ for MergeSort run on an array of size n is

$$h = \lceil \lg n \rceil. \quad (\text{A3})$$

Proof. $T(n)$ has n leaves (one for each of the n 1-element subarrays of the sorted array that are passed to (recursive) calls to MergeSort). Since every level k of $T(n)$, except, perhaps, for the last level h , has the maximal number 2^k of nodes that any binary tree can have at level k , a binary tree with n leaves could not be any shorter than $T(n)$. Hence, $T(n)$ is a shortest binary tree with n leaves. Therefore, (by a well-known fact—see, for instance, [2], page 68, Lemma 2.5), its depth h is equal to $\lceil \lg n \rceil$. Thus the equality (A3) holds. $\square$

Because each node in any level above $h - 1$ shows a value ≥ 2 , it has 2 children. Thus the value it shows is equal to the sum of values shown by its children, as I have indicated at the beginning of this Section. From that, I conclude (a proof by induction is left as an exercise for the reader) that the sum of values shown at nodes in any level $i = 0, \dots, h - 1$ is the same for each such level. Thus the said sum is equal to the value shown by the only node at level 0, that is, equal to n .

Let $a_1, \dots, a_{2^i}$ be the values shown at the nodes of some level $i = 0, \dots, h - 1$. The number of comps performed by a call to Merge invoked by the call to MergeSort on an array of a_j elements is either 0 if $a_j = 1$ (no call to Merge is made) or, as I have shown in the previous section, is $a_j - 1$ if $a_j \geq 2$. Therefore, in either case, it is $a_j - 1$. Thus the number of comps C_i performed at level i is

$$C_i = (a_1 - 1) + \dots + (a_{2^i} - 1) = (a_1 + \dots + a_{2^i}) - \underbrace{(1 + \dots + 1)}_{2^i} = n - 2^i. \quad (\text{A4})$$

Moreover, since all nodes at the last level h are 1s,

$$C_h = 0. \quad (\text{A5})$$

Therefore, the total number $W(n)$ of comps that MergeSort performs in the worst case on an n -element array is equal to

$$W(n) = \sum_{i=0}^h C_i =$$

[by (A5)]

$$= \sum_{i=0}^{h-1} C_i =$$

[by (A4)]

$$= \sum_{i=0}^{h-1} (n - 2^i) = nh - (2^h - 1) = nh - 2^h + 1 =$$

[by (A3)]

$$= n \lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1.$$

This way, I have proved the following.

Theorem A3. The number $W(n)$ of comparisons of keys that MergeSort performs in the worst case while sorting an n -element array is

$$W(n) = n \lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1. \quad (\text{A6})$$

Proof. Follows from the above derivation. $\square$

Using the well-known (see, for instance, [3]) closed-form formula for $\sum_{i=1}^n \lceil \lg i \rceil$, I conclude that

$$W(n) = \sum_{i=1}^n \lceil \lg i \rceil. \quad (\text{A7})$$

Appendix B. Proof of $\sum_{i=0}^{m-1} \lfloor \frac{n+i}{m} \rfloor = n$

Theorem A4. For every natural number n and every positive natural number m ,

$$\sum_{i=0}^{m-1} \lfloor \frac{n+i}{m} \rfloor = n.$$

Proof. Let $n = km + l$, where $0 \leq l < m$.

We have

$$\lfloor \frac{n+i}{m} \rfloor = \lfloor \frac{km+l+i}{m} \rfloor = \lfloor k + \frac{l+i}{m} \rfloor = k + \lfloor \frac{l+i}{m} \rfloor.$$

Therefore,

$$\sum_{i=0}^{m-1} \lfloor \frac{n+i}{m} \rfloor = mk + \sum_{i=0}^{m-1} \lfloor \frac{l+i}{m} \rfloor = mk + \sum_{i=m-l}^{m-1} \lfloor \frac{l+i}{m} \rfloor = mk + \sum_{i=m-l}^{m-1} 1 = mk + l = n.$$

$\square$

References

1. Sedgewick, R.; Flajolet, P. *An Introduction to the Analysis of Algorithms*; Pearson: Boston, MA, USA, 2013.
2. Baase, S. *Computer Algorithms: Introduction to Design and Analysis*, 2nd ed.; Addison-Wesley Publishing: Reading, MA, USA, 1991.
3. Knuth, E.K. *The Art of Computer Programming*, 2nd ed.; Addison-Wesley Publishing: Reading, MA, USA, 1997; Volume 3.
4. Trollope, J.R. An explicit expression for binary digital sums. *Math. Mag.* **1968**, *41*, 21–25. [\[CrossRef\]](#)
5. McIlroy, M.D. The number of 1's in binary integers: Bounds and extremal properties. *SIAM J. Comput.* **1974**, *3*, 255–261. [\[CrossRef\]](#)
6. Allouche, J.-P.; Stipulanti, M. Summing the sum of digits. *Commun. Math.* **2024**, *33*, 2. [\[CrossRef\]](#)
7. Knuth, D.; Graham, R.; Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science*; Addison-Wesley: Reading, MA, USA, 1994.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.