Precise Evaluation of Execution Cost of Sequence Rotation by Three-Way-Reversals

Joseph A. ERHO1 and Juliana I. CONSUL2

Niger Delta University,
Mathematics/Computer Science Department, Nigeria

IJASR 2019
VOLUME 2
ISSUE 5 SEPTEMBER - OCTOBER **ISSN: 2581-7876**

Abstract – Sequence blocks exchange or rotation often requires greatest common divisor (gcd) calculated or three-way-reversal swaps of elements to accomplish it. Using three way sequence reversal has somewhat posed challenges in effort to give exact number of index comparisons and elements moves. This is commonly seen with the use of terms or operators like "about", "≤", or "Δ" to indicate approximate or possible number of comparisons and/or assignments. In this paper, we propose concise way of expressing the precise and exact number of comparisons and thus, number of moves in the three-way-reversal based rotation using simple equation. With five input scenarios, we show how the equation produces precise values and give reason why previous studies had to rely on approximation. The current study is useful in complexity analysis of algorithm such as in-place merging in sorting. The data considered, with their computed outcome, quickly suggest the need for further work on gcd based rotation.

Keywords: Sequence block exchange, sequence rotation, sequence reversal, gcd cycle rotation, rotation complexity analysis, algorithm, equation.

Introduction

While merging subsequences in 'in-place' sorting algorithms, we often encounter the leading choices of gcd calculated exchange of blocks or three way reversal of the input [2] among others [4]. Though experts tend to favour the use of gcd calculated exchange of blocks [5,7, 8, 9] , it is increasingly common to see many using three stages reversal approach [6, 9,11]. Experts believe that gcd calculated exchange of blocks is more efficient than three way reversal [3]. This argument is usually based on smaller number of elements comparisons with small sizes of indexes involved, which speeds up [10]. But this is not always true.

For instance, a live database table comprising of over seven millions students exams scores, generated over a period of time was to be analysed. The records per course, per student, per class, per programme, etc were to be sorted by the scores key before analysis. The scores ranging from -1 to 100 were small enough values for comparison as against indexes that range in the millions. Items comparisons and moves in this case were trivial, as against indexes comparisons and moves. *So, the efficiency of gcd calculated rotation of sequence blocks would depend on the size of individual items, number of items, and/or the range of involved indexes in the comparison and moves*. Besides, gcd calculated block exchange is more complicated [11]. Therefore, it is not surprising to see increasing interest on three-way-reversal based rotation. It is easy and more "effective" method in practice [4] and considered to be folklore.

Unfortunately, those who are in favour of three way reversal for block rotation hardly give a precise value for the execution complexity cost for it [10, 11]; and sometimes even erroneous values are presented [2]. Typically, while [10] accurately determined the number of items moves to be *"...exactly* $\left\lfloor \frac{j-i+1}{2} \right\rfloor$ *swaps."* for "$INVERSE(i,j)$" (a culture we would want to uphold), its block exchange counterpart could not have a fixed precision decomposed result that should naturally arise from already calculated exact values, where "$INVERSE(i,j)$" is inverse function on sequence "segment" spanning i to j. Rather, it is computed as " $BLOCK\_EXCHANGE\_1(c, l_1, l_2) = \cdots = \left\lfloor \frac{l_1}{2} \right\rfloor + \left\lfloor \frac{l_2}{2} \right\rfloor + \left\lfloor \frac{l_1+l_2}{2} \right\rfloor \leq l_1 + l_2$" - an infinite range, bounded above by $(l_1 + l_2)$, where c, l_1 and l_2 are the start of sequence "segment", sizes of first and second subsequences, respectively. Interestingly, the same author in [10], Symvonis, reviewed it later "... *that about* $l_1 + l_2$ *swaps are performed...* " [11]. We note that the use of "***about***" obviously lack fixed result that should come from addition of exact values. It becomes a little more worrisome

when [2] concluded that the rotation "...*uses a total of* $u + v - \Delta$. *Where* $\Delta = 0$ *if both* u *and* v *are even, 1 if either but not both* u *or* v *is odd, 2 if both* u *and* v *are odd...*". The conditions that Δ equals 1 or 2 are erroneous, as will be shown in section 4.

In contrast, gcd based block rotation is often given a precise value for number of moves of items. It is usually given as $g\left(\frac{|UV|}{g} + 1\right) = |UV| + \gcd(|U|, |V|)$ [1]. Here, $g = \gcd(|U|, |V|)$ represents the number of cycles required to perform the rotation task while U and V are two blocks. Even more precision is given to it as:

> "... *can be reduced to* $|U|+|V|+1$ *if the order of elements in the larger-sized one of* U *and* V *is kept unchanged, and* $2\min(|U|; |V|) + 1$ *if the order of elements in the smaller-sized one of* U *and* V *is kept unchanged*" [1]

though, number of index comparison is not included. Thus, it is desirable that a ***precise computation be given to the number of moves and index comparisons of a three way inversion based rotation requires***. The intent is to provide sorting algorithm analysts a simple and concise equation that enables them analytically give precise results for given inputs. The study is also aimed at abolishing the inconsistencies of results inherent with the use of existing method for evaluating sequence rotation by three-way-reversal.

In this research a numerical evaluation method in five different input scenarios are used to show the correctness of the equation and pictorial illustration are used when necessary to clarify ideas.

In the rest of this paper, we express the rudiments of sequence reversal in section 2, which is the working tool for rotation. Section 3 discuses the fundamentals of sequence rotation and states our proposed equation. We prove this numerically in section 4, using the results to show the correspondence between the usual evaluation and our new method. Section 5 analyses and discusses the results of the numerical evaluation, explaining the root cause of the inconsistencies presented in the literature. We conclude in section 6 with some direction for further works.

Sequence Reversal

A sequence ***reversal*** *(or* ***inversion****) is simply a consecutive swapping of opposite items, from both ends of the sequence, until the mid item is met.* If sequence $A = \{(a_i)_{i\in\mathbb{N}}: a_i \le a_{i+1}\}$, then $A^{-1} = \{(a_j)_{j\in\mathbb{N}}: a_j \ge a_{j+1}\}$ would be the reverse/inverse of A. Notice that the set A has single item; and that single item is a sequence of elements defined by the predicate of the set. For example, to reverse (or compute reversal for) a sequence $A = \{(a, b, c, d, e, f)\}$, we simply swap or exchange opposite items a and f, b and e, and the mid items c and d to obtain $A^{-1} = \{(f, e, d, c, b, a)\}$. Where only a single item is remaining (at the middle) a valid pair cannot be constituted. And it is worthless to swap an element with itself.

The reversal computation requires a space complexity of $O(1)$ i.e. a constant one memory space to keep single temporary value for the swap. Hence, it is an ***in-place*** merging process. Also, it requires $\left\lfloor\frac{n}{2}\right\rfloor + 1$ comparison of index, zero (0) comparison of items, and $3\left\lfloor\frac{n}{2}\right\rfloor$ moves of items. This is typical $O(n)$ complexity, with high efficiency since no items comparison is required. For instance, from the sequence A above, index of item a been compared with index of item f is counted as one (1) comparison and that of b with e is another. So for this sequence with 6 items would have $\left\lfloor\frac{6}{2}\right\rfloor + 1 = 4$ index comparisons. Note that the +1 is the last comparison (i.e. exit test) that terminates the loop and thus not included in the number of moves. We adopt the Visual Studio C++ sequence reversal (or inversion) and apply it 3 time for ***rotation***.

Sequence Rotation

By ***rotation*** of items at positions p to r about q to $(q + 1)$ pivot (i.e. $(q, q + 1)$ ***the pivot***) [12], we mean 3 consecutive stages of reversals: (1) reversing subsequence p to q, (2) subsequence $(q + 1)$ to r, and finally, (3) sequence p to r. *The effect is like holding subsequence spanning positions* p *to* q *at point* q *and sliding the* q *end to* r*, then sliding that of* $(q + 1)$ *to* p*, making* $(q, q + 1)$ *the pivot*. Now the original positions of p and r can overlap at the original positions of q and $(q + 1)$, depending on the length of p to q and $(q + 1)$ to r. *(See the Figure 1)*

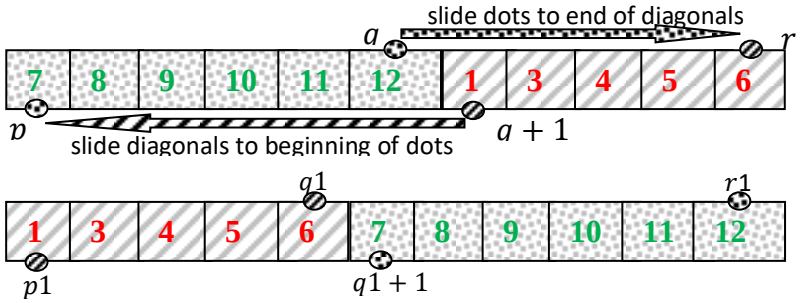

Figure 1: sliding effect of sequence rotation

By extension of reversal complexity, sequence *rotation* of $m+n$ items requires exactly

$$\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor + 3 \equiv m + n + 3 - ev, \qquad (1)$$

where $ev = \begin{cases} m \bmod 2, & if \ m \bmod 2 = n \bmod 2 \\ 1, & otherwise \end{cases}$

comparison of indexes (ev stands for erhova), zero (0) comparison of items, and

$$3\left(\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor\right) = 3(m + n - ev) \qquad (2)$$

moves of items. Here, we have introduced a function ev that enforces the precision on the number of comparison and moves. In section 4, we show five scenarios that prove equations (1) and (2) correct. First, we use Figure 2 below to illustrate that the number of index comparisons is actually $\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor + 3$ leading to exactly $3\left(\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor\right)$ elements moves. This move expression is common in the literature [6, 9,11].

In Figure 2, sequence (a) is rotated using (b), (c) and (d) stages reversals to obtain (e)

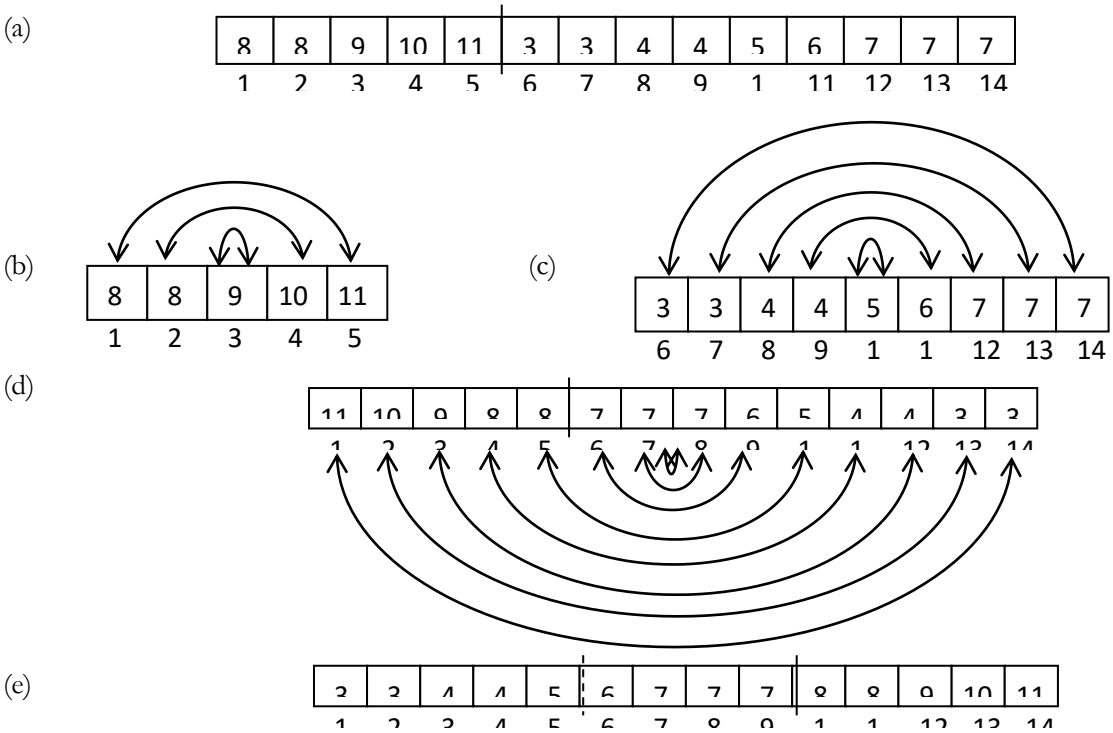

Figure. 2: (a) the sequence to rotate, (b) reversal of left subsequence, (c) reversal of right subsequence, (d) reversal of the resulting sequence of the previous two stages, (e) the final result of rotation.

From the diagram, we rotate the sequence from index 1 to 14 about index (5,6) pivot using three-stage reversals. Arrow headed arcs are used to show pair of indexes that are comparable. We want to establish a convention that *there must be two distinct valued indexes making a pair for comparison to be valid and/ or efficient.* (*See previous section*) That is, given two subsequence $A \in X$ and $B \in X$, with item $a_i \in A$ and $b_j \in B$, where $X = (A, B)$, indexes i and $j, i \neq j$ for items forming a pair, $i < j$ *constitutes a valid and/or efficient comparable pair.* Any time inefficient comparison is made the process must terminate immediately without elements assignment. Observe that the innermost arcs point to invalid comparable pair. They simply mark the last tests that terminate the loops and obviously add up to the number of index comparisons.

Numerical Evaluation

From Figure 2, the first 5 indexes comparison has 3 arcs, equivalent to $\left\lfloor \frac{5}{2} \right\rfloor + 1$, the second set of 9 index has 5 arcs corresponding to $\left\lfloor \frac{9}{2} \right\rfloor + 1$, and finally, overall reversal of all 14 items use 8 arcs equivalent to $\left\lfloor \frac{14}{2} \right\rfloor + 1$ comparisons. The addition of 1 to each reversal stage corresponds to the innermost arcs that points to invalid comparable pair - singleton. So rotation by reversal uses exactly $\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor + 3$ index comparisons. As such, it uses exactly $3\left(\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor\right)$ elements moves. Notice the disappearance of the constant 3 in both expressions for data moves. It represents the three looping termination index tests. Next, we show our equivalent expression in the comparison equation $\left\lfloor \frac{m}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{m+n}{2} \right\rfloor + 3 = m + n + 3 - ev$ using the five scenarios.

Scenario 1: both m and n are odd but $m \neq n$,

Let $m = 5$ and $n = 9$. From equation (1),

$\left\lfloor \frac{5}{2} \right\rfloor + \left\lfloor \frac{9}{2} \right\rfloor + \left\lfloor \frac{5+9}{2} \right\rfloor + 3 = 2 + 4 + 7 + 3 = 16.$

Equivalently, $m + n + 3 - ev = 5 + 9 + 3 - ev = 17 - ev$.

Now $ev = \begin{cases} m \bmod 2, & if \ m \bmod 2 = n \bmod 2 \\ 1, & otherwise \end{cases}$

and $m \bmod 2 = 5 \bmod 2 = 1$, also $n \bmod 2 = 9 \bmod 2 = 1$

So, since $m \bmod 2 = n \bmod 2$, then $ev = m \bmod 2$, i.e. $ev = 5 \bmod 2 = 1$

thus $m + n + 3 - ev = (5 + 9 + 3) - 1 = 17 - 1 = 16$ ■

Scenario 2: both m and n are odd and $m = n$,

Let $m = 7$ and $n = 7$. From equation (1)

$$\left\lfloor \frac{7}{2} \right\rfloor + \left\lfloor \frac{7}{2} \right\rfloor + \left\lfloor \frac{7+7}{2} \right\rfloor + 3 = 3 + 3 + 7 + 3 = 16$$

and since $m \bmod 2 = n \bmod 2 = 1$, $m + n + 3 - ev = (7 + 7 + 3) - 1 = 17 - 1 = 16$ ■

Scenario 3: m is odd but n is even or vice versa,

Let $m = 5$ and $n = 8$. From equation (1)

$$\left\lfloor\frac{5}{2}\right\rfloor+\left\lfloor\frac{8}{2}\right\rfloor+\left\lfloor\frac{5+8}{2}\right\rfloor+3=2+4+6+3=15$$

Observe that even though only m or n of the two is odd, the dual-odd-values phenomenon surfaces in the combined $\left\lfloor\frac{5}{2}\right\rfloor$ and $\left\lfloor\frac{5+8}{2}\right\rfloor$. Now, since $m \bmod 2 \neq n \bmod 2$, then $ev = 1$.

Equivalently, $m+n+3-ev=(5+8+3)-1=16-1=15$.

Scenario 4: both m and n are even and $m=n$,

Let $m=8$ and $n=8$. From equation (1)

$$\left\lfloor\frac{8}{2}\right\rfloor+\left\lfloor\frac{8}{2}\right\rfloor+\left\lfloor\frac{8+8}{2}\right\rfloor+3=4+4+8+3=19$$

now, $m \bmod 2 = n \bmod 2$, Interestingly $m \bmod 2 = 0$ and so $ev = 0$

Equivalently, $m+n+3-ev=(8+8+3)-0=19-0=19$

Scenario 5: both m and n are even but $m \neq n$,

Let $m=30$ and $n=56$. From equation (1)

$$\left\lfloor\frac{30}{2}\right\rfloor+\left\lfloor\frac{56}{2}\right\rfloor+\left\lfloor\frac{30+56}{2}\right\rfloor+3=15+28+43+3=89$$

now, $m \bmod 2 = n \bmod 2$, $m \bmod 2 = 0$ and so $ev = 0$.

Equivalently, $m+n+3-ev=(30+56+3)-0=89-0=89$

Discussion

From the numerical solutions, it was clear that $\left\lfloor\frac{m}{2}\right\rfloor+\left\lfloor\frac{n}{2}\right\rfloor+\left\lfloor\frac{m+n}{2}\right\rfloor+3\equiv m+n+3-ev$. An interesting conclusion about this proof was that whenever both m and n were even, the value of ev (erhova) was always 0 but 1 otherwise. Also, ev had exact value that depended on m and n.

The ev value evolved from the floor function that truncated the fractional part of a term division. Whenever at least one of m and n was odd, exactly two terms of $\left\lfloor\frac{m}{2}\right\rfloor$, $\left\lfloor\frac{n}{2}\right\rfloor$, and $\left\lfloor\frac{m+n}{2}\right\rfloor$ generated fractions (precisely 0.5 each) to be discarded. Each of these two fractions indicated a single remaining index that needed not be compared in their respective subsequences. These singular index from each of the two terms would constitute a valid comparable pair of indexes, but never duly participated in their respective subsequence index comparisons. The non involvement of this invalid pair in comparison demands that we subtract it from the overall number of valid pair comparison and the constant 3. This is what is being determined by the value $ev=\begin{cases} m \bmod 2, & if\ m \bmod 2 = n \bmod 2 \\ 1, & otherwise \end{cases}$

This fact has not been properly considered in previous studies, but merely relied on approximate values. Such invalid pair did not exist though if both m and n were even. As such $ev=0,\ \forall\, m,n$, irrespective of whether $m=n$ or not.

Now, where both left and right subsequences are equal in length (i.e. $w=2m$), a more efficient way is not to use three-way-reversal rotation. We simply apply direct swapping of corresponding items on both sides, resulting in just $\frac{w}{2}$+1 comparison of indexes, zero (0) comparison of items and $3\frac{w}{2}$ moves of items. So, complexity of the rotation is $O(w)$, where w is the sum of sizes of both left and right. It could be seen immediately that applying

direct swapping of corresponding items, if both sides are equal, is far more efficient than three-stage reversal of input.

Preliminary Conclusion

This paper presented a simple and concise equation that enables sorting algorithm analysts determines precisely the complexity cost of sequence rotation by reversal for given inputs. With precise complexity evaluation, inconsistencies of results inherent with old methods can now be abolished.

It has been shown that rather than having approximate or range of values for complexity cost of rotation by three-way-reversal, we can actually compute precise value for given inputs. The reason the existing methods fails to give fixed result was also discussed and to fix that we introduced the function ev into the equation.

With the accuracy of the results, we hope to investigate also the complexity of gcd based rotation on data (such as the data case in section 1) that have sizes much smaller than sizes of indexes and those whose elements and indexes sizes average out; since the common assumption has always been 'index is small as compare to item sizes'.

References

1. Chen J. (2003) Optimizing stable in-place merging, Theoretical Computer Science, VOL. 302, 191–210 https://core.ac.uk/download/pdf/82464674.pdf (Accessed 4 Sept 2019)
2. Coates-Evelyn D. (2004) In-Place Merging Algorithms, Technical Report, Department of Computer Science, King's College London str. 1-46 https://nms.kcl.ac.uk//informatics/techreports/papers/TR-04-05.pdf (Accessed August 29, 2019)
3. Dudzinski, K., Dydek, A. (1981) On a stable minimum storage merging algorithm, Information Processing Letters, VOL. 12, 5
4. Furia C. A. (2015) Rotation of Sequences: Algorithms and Proofs, ETH Zurich, Switzerland https://arxiv.org/abs/1406.5453 (Accessed 29 Aug 2019)
5. Geffert V., Katajainen J., Passanen T. (2000) Asymptotically Efficient In-Place Merging, Theoret. Comput. Sci., VOL. 237, 159-181
6. HUANG't B-C. and LANGSTON M. A. (1992) Fast Stable Merging and Sorting in Constant Extra Space, The Computer Journal, VOL. 35, No 6, Page 643-650
7. Kim P., Kutzner A. (2006) On Optimal and Efficient in Place Merging SofSem, LNCS 3831, pp.350- 359
8. Kim P. and Kutzner A.(2008) Ratio based stable in-place merging http://itbe.hanyang.ac.kr/ak/papers/tamc2008.pdf (Accessed 29 Aug 2019)
9. Mohammed J. L. and Subi C. S. (1987) An Improved Block-Interchanged algorithm, VOL. 8, 113-121
10. Symvonis A.(1994) Optimal Stable Merging., Int. Conf. on Computing and Information, Proc. ICCI' VOL. 94, 124-143 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.4103&rep=rep1&type=pdf (Accessed 31Aug 2019)
11. Symvonis A.(1995) Optimal Stable Merging, The Computer Journal, Volume 38, Issue 8, Pages 681–690, https://doi.org/10.1093/comjnl/38.8.681
12. Xinok (2014) In-Place Merge Sort Demystified https://xinok.wordpress.com/2014/08/17/in-place-merge-sort-demystified-2/ (Accessed 29 Aug 2019)