Poor Performance of Juggling Sequence Rotation as Greatest-common-divisor Dependent

Joseph Agaroghenefuoma Erho, *Member, IAENG,* Bunakiye Richard Japheth, *Member, IAENG,*
Evans Fiebibiseighe Osaisai and Juliana Iworikumo Consul

***Abstract*—A previous study of two algorithms performances in a given practical experiment using Java for four different data types on a sequence of 20,000,000 elements showed striking results. Focusing on type LONG, for instance, the average execution times showed that, three-way-reversal sequence rotation took only 49.66761*ms*, whereas that of juggling rotation was as big as 246.4394*ms* - approximately 496.18% difference! This current study is therefore set out to investigate the actual source of this huge up surge in the average execution time of the juggling sequence rotation. Data were extracted from a large pool of the previous study, presented in a different format and analyzed. The current study shows that the juggling rotation performance for some cycle gaps actually competes favorably with three-way-reversal rotation (and even outperforms the latter for certain cycle gaps), but performs poorly with some other consecutive cycle gaps. This study observes that the poor performance was worst and enormous with cycle gaps that were around and closest to the *square root* of the sequence size. The current study therefore advises application developers to be mindful of the use of juggling rotation, if cycle gaps are around the square root of sequence size and, in particular, if the application is time critical. Concentrating only on scenario of likely best performance behavior of the juggling rotation, this study is restricted to greatest common divisors values that are equal to cycle gaps.**

***Index Terms*—GCD rotation, juggling algorithm, juggling rotation, sequence exchange, sequence rotation.**

I. Introduction

GREATEST common divisor (GCD) has been used in juggling algorithm to exchange (or rotate) block or section of sequence with another in an *in-place* sorting [1], [2], internal buffer management in text editors [3] (see also [4]). It had been determined that GCD based rotation or "*vector exchange*" [5] is more efficient than some other rotation algorithms [6], because "*the problem solution can be reduced*" [7], [8] to $m+n+gcd(m+n,n)$ moves [9] as against its three-way-reversal counterpart that used as much as $3(\lfloor(m+n)/2\rfloor+\lfloor m/2\rfloor+\lfloor n/2\rfloor)$ moves [5], where $m+n$ is the sequence size and n as the cycle gap.

It was also derived by [7] , using least common multiple (LCM) and GCD [10], that GCD based rotation actually requires $m+n+gcd(m+n,n)$ elements moves and $2m+3n-gcd(m+n,n)$ index moves for some implementation style (see [9] for explicit GCD calculated rotation). Yet, the study in [7] showed that, despite sizable number of elements moves $3(m+n-ev)$, and smaller indexes moves $m+n+3-ev$, of three-way-reversal ($ev=0$ or 1, due to [11]), there is significantly huge discrepancy between the average execution time of the GCD rotation and the three-way-reversal approach across all four data types of LONG, INT, SHORT, and CHAR in Java.

Taking one of the data type, LONG for instance, [7] showed that, on the average, execution time for three-way-reversal was only 49.66761*ms*, whereas that of GCD based rotation was as huge as 246.4394*ms*! What could contribute to this apparent big gap, given the theoretical derivation that GCD based rotation used only $m+n+gcd(m+n,n)$ elements moves as against three-way-reversal rotation that used as much as $3(m+n-ev)$ elements moves? The interest of this current study is therefore ***to investigate the source of the huge up surge in the average execution time of the GCD based rotation***. The aim is to give a better understanding of the behavior of GCD based rotation in order to achieve an objective of guiding implementer of the juggling rotation. The study is limited to GCD's that are equal to cycle gaps, since these appear to give juggling rotation better performance. Even though there are other implementation styles [12], [13], including non-GCD explicit version in [9], this study is restricted to the GCD based implementation presented in [9]. The following notations are used in this article: **gcd-R** – GCD based rotation; **twr** – three-way-reversal rotation.

In this paper, section I contains the introduction that clearly states the problem. Section II highlights the tools and methods used in the experimentation. Section III presents and analyses the results. In discussing the results section IV explains the implication of the outcome. Finally section V concludes the paper, giving some direction for future scope. We begin with the tools used and the way experiment was carried out to achieve our results.

II. Materials and Methods

The data presented in this study were extractions from the ones studied in [7] but distinctively presented in different forms. So the tools used were exactly the ones described in the previous study. That study evaluated the performance of juggling and three-way-reversal algorithms over LONG, INT, SHORT, and CHAR data types in Java - testing to see whether index size affects the performance. However, this current study did not give attention to index moves execution

Manuscript received November 5, 2020; revised April 28, 2021.

J. A. Erho is with the Department of Computer Science, Niger Delta University, Wilberforce Island, P.M.B. 071, Amassoma, Bayelsa State, Nigeria. e-mail: joseph.erho@mail.ndu.edu.ng.

B. R. Japheth is with the Department of Computer Science, Niger Delta University, Wilberforce Island, P.M.B. 071, Amassoma, Bayelsa State, Nigeria. e-mail: bunakiye.japheth@ndu.edu.ng.

E. F. Osaisai is with the Department of Mathematics, Niger Delta University, Wilberforce Island, P.M.B. 071, Amassoma, Bayelsa State, Nigeria. e-mail: fevansosaisai@gmail.com

J. I. Consul is with the Department of Mathematics, Niger Delta University, Wilberforce Island, P.M.B. 071, Amassoma, Bayelsa State, Nigeria. e-mail: ji.consul@ndu.edu.ng.

timing, since the concern here was on influence of cycle gaps / GCD on execution timing. Also, the sampled data used here were selection of only one record per cycle gap instead of five (each from every next 100 records) of the previous study.

In the current study, extract of 71 records, representing the different cycle gaps, from the data of [7] are presented below. Table I is extracted from the LONG data type records sample. The table is arranged in such a way that several cycle gaps records can be viewed side by side. The table-page columns should be seen as sequential. For example, table-page 2 column follows table-page 1 column, in that order. The serial number (S/N) column of the rows are used to number the 71 cycle gaps and represents same in the horizontal entries of the graphs. On the table, serial numbers 1 to 5, for example, represent cycle gaps 1, 2, 4, 5, 8 and serial numbers 69, 70, 71 represent 400000, 500000, and 10000000 respectively. The table is arranged in descending order of both the cycle gaps and the serial numbers in line with the running of the algorithm code on the cycle gap values. Sequel to this, the horizontal entries of the graph for serial number 1, 3, and 5 represent the cycle gaps 1, 4, and 8 respectively. Although, three-way-reversal rotation behavior is not given separate analysis, its data are placed alongside the juggling rotation for an easy comparison of the deviations from uniform efficient performance. What are our findings?

III. RESULTS

A quick look through the first few cycle gaps in the table of rotation execution timing shows that GCD based rotation actually performs better than three-way-reversal rotation. Yet, reference [7] successfully showed that, on the average, execution time for three-way-reversal is only 49.66761ms whereas that of GCD based rotation is as big as 246.4394ms! What could be reason for this apparent big gap?

A. Analysis Of The Graphs

Though fig. 1 is not really necessary, since it represents the table I, but we present it here for the purpose of easily comparing it with the other figures [14]. The three-way-reversal rotation graph is plotted alongside the juggling rotation for a clear comparison and a feel of the deviations from efficient performance. Plotting the extracted data in Excel, the look of all four graphs tends to be like bell shapes. This means that GCD rotation efficient performance is lowest around a consecutive range of cycle gaps, quite some points away from both ends and concentrating around the middle of the cycle gaps listing. For data type LONG, the range is between point 9 and 44 representing cycle gaps 25 to 16,000. But notice the high performance indication from point 45 to 65, representing cycle gaps 20,000 to 1,000,000 and even outperformed three-way-reversal rotation at point 66 to 71. This trend is similar with those of integer data type *(see fig. 1 and 2 graphs)*.

The case is a bit different for short and character data types. With these types the bell shapes are still visible but with some sort of down skewness close to the peak of the bells, tending to split the single bells into two instead, respectively. This occurs around point 18 to 31, representing cycle gaps 160 to 1600. In similarity with types LONG and INT, the SHORT and CHAR also have high performance indication from point 1 to 5, representing cycle gaps 1 to 8 and from point 50 to 71, representing cycle gaps 50,000 to 10,000,000. Type char even have higher performance indication from point 1 to 3, representing cycle gaps 1, 2, and 4 *(see fig. 3 and 4 graphs)*.

IV. DISCUSSION

In the previous study by reference [7], it was shown that GCD based rotation demonstrated, on the average, a huge up surge of 246.4394ms execution timing as against three-way-reversal time of just 49.66761ms. This was approximately 496.18% difference. What this current research has shown is that the GCD rotation performance for some of the cycle gaps actually compete favorably with three-way-reversal rotation - that is, maintaining a good and balanced efficiency in performance. This can easily be seen from the two extremes of all four graphs.

Surprisingly though, the plot suddenly jumped up after few cycle gaps and then down before some few remaining cycle gaps. The pattern produced appears to be bell shaped. This could not have been mere outliers, otherwise it would not have maintained somewhat uniform pattern across all four graphs. The conclusion reachable here is that there could be a phenomenon playing out - for some cycle gaps (close to 1 downwards and close or up to half of the sequence) the performance is quite efficient but very poor otherwise.

A careful look at table 1 shows that the peak of this poor performance is with cycle gaps that are around the square root of the sequence size and all four graphs conform to this fact. For instance, the sequence size used for this experiment was 20,000,000 and the square root was approximately 4,472.14. The table has it that about seven (7) cycle gaps to both left and right of the *square root* value (4,472.14) fall into the peak poor performing values for the GCD based sequence rotation. This range of cycle gaps corresponds to the range 30 to 43 horizontal entries for the graphs. This appears to cut across all four data types as can be seen from the graphs.

Some form of skewness is also observed depending on the data types. It appears that as the data type decreases from long (LONG) to character (CHAR), the skewness becomes more and more pronounced. This pattern tends to split the single bell apparently into two in the case of SHORT and CHAR types. These points of skewness tend to improve the GCD rotation performance a little. Next is concluding remarks.

V. CONCLUSION

This paper has analyzed the relationship between sequence cycle gaps, which was used to compute the GCD, and the performance of GCD based rotation. Using randomly sampled data, extracted from [7] but presented in a different format, the study found that efficient performance of the algorithm depended on the cycle gap, which determined GCD. For cycle gaps that were close to 1 downwards and/or close or up to half of the sequence, the performance was quite efficient.

Perhaps, this might explain the reason some previous studies quickly concluded that GCD rotation performed

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
SAMPLED DATA FOR GCD ROTATION AND THREE-WAY-REVERSAL ROTATION, ONE RECORD PER CYCLE GAP OF 71 *(see [7])*.

Table Page 1				Table Page 2				Table Page 3				Table Page 4			
gcd-R	twr	cycle gap	S/N	gcd-R	twr	cycle gap	S/N	gcd-R	twr	cycle gap	S/N	gcd-R	twr	cycle gap	S/N
45	49	10000000	71	133	49	80000	53	464	50	3200	35	270	48	128	17
43	48	5000000	70	144	53	78125	52	437	51	3125	34	281	48	125	16
43	48	4000000	69	78	49	62500	51	448	59	2500	33	284	49	100	15
44	51	2500000	68	99	49	50000	50	405	50	2000	32	303	48	80	14
50	48	2000000	67	105	52	40000	49	427	50	1600	31	270	48	64	13
48	51	1250000	66	212	49	32000	48	440	48	1280	30	257	48	50	12
61	52	1000000	65	123	49	31250	47	397	49	1250	29	305	49	40	11
131	54	800000	64	102	50	25000	46	376	48	1000	28	315	49	32	10
74	49	625000	63	152	50	20000	45	390	48	800	27	321	51	25	9
69	59	500000	62	407	48	16000	44	398	50	640	26	292	48	20	8
108	52	400000	61	379	49	15625	43	382	52	625	25	272	49	16	7
84	48	312500	60	406	49	12500	42	411	48	500	24	207	50	10	6
67	48	250000	59	490	48	10000	41	376	50	400	23	173	49	8	5
99	47	200000	58	433	47	8000	40	381	49	320	22	106	48	5	4
195	49	160000	57	479	50	6400	39	303	47	256	21	89	49	4	3
113	57	156250	56	469	49	6250	38	347	49	250	20	54	49	2	2
66	51	125000	55	466	48	5000	37	309	49	200	19	43	48	1	1
116	51	100000	54	462	49	4000	36	310	49	160	18				

better than three-way-reversal rotation. Mostly in an in-place sorting, the block to rotate with another would likely be about half of the sequence under consideration. In such a case, the implementation simply fell in the range of cycle gaps that were *GCD rotation friendly* - compare fig. 4 and the data type used in [9].

The study found that the situation was quite different when the cycle gaps were not in the ranges of GCD rotation friendly. In fact, the poor performance was peak when the algorithm used cycle gap that was within about seven (7) cycle gaps to both left and right of the *square root* of the given sequence size. However, some form of skewness was also observed that depended on the data types. SHORT and CHAR types in particular, exhibited this, tending to split the single bells shaped pattern of GCD rotation into two respectively. These points within the splitting region showed slight improvement from the peak poor performance of the algorithm.

Thus, the GCD based sequence rotation efficient performance heavily depends on the *greatest-common-divisor* which is a direct reflection of the cycle gaps. This research will therefore strongly recommend that if application involves sequence rotation with cycle gaps that are within about *seven (7) cycle gaps* to both left and right of the *square root* of the given sequence size, DO NOT use GCD based sequence rotation. For applications that are time critical, DO NOT use juggling rotation if the cycle gaps are not within *GCD rotation friendly* range.

It is clear from this study that cycle gaps around the square root of sequence size are heavily slow in juggling rotation. The question is why is fewest number of elements assignment not proportionate with execution time of juggling rotation? There is need for further work on this.

REFERENCES

[1] B.-C. Huang and M. A. Langston, "Practical in-place merging," *Communications of the ACM*, vol. 31, pp. 348–352, 1988.

[2] X. Wang, Y. Wub, and D. Zhua, "A new variant of in-place sort algorithm," in *Proceedings of the International Workshop on Information and Electronics Engineering IWIEE (10-11 March 2012, Harbin, Heilongjiang)*. China: Elsevier, 2012, pp. 2012, 2274–2278.

[3] A. A. Stepanov and D. E. Rose, *From Mathematics to Generic Programming*. USA: Pearson Education Inc, 2015.

[4] J. Bentley, *Programming Pearls*, 2nd ed. USA: ACM Press/Addison-Wesley Inc, 2000.

[5] K. Dudzinski and A. Dydek, "On a stable minimum storage merging algorithm," *Information Processing Letter*, vol. 12, pp. 5–8, 1981.

[6] P. S. Kim and A. Kutzner, "Ratio based stable in-place merging," in *Proceedings of 5th International Conference on Theory and Applications of Models of Computation (25-29 April 2008, Xi'an, China)*, M. Agrawal, D. Z. Du, Z. Duan, and A. Li, Eds. China: Springer, 2008, pp. 2008, 246–257.

[7] J. A. Erho, J. I. Consul, and B. R. Japheth, "Juggling versus three-way-reversal sequence rotation performance across four data types," *International Journal of Scientific Research in Computer Science and Engineering*, vol. 7, pp. 10–18, 2019.

[8] A. Symvonis, "Optimal stable merging," *The Computer Journal*, vol. 38, pp. 681–690, 1995.

[9] C.-K. Shene, "An analysis of two in-place array rotation algorithms," *The Computer Journal*, vol. 40, pp. 541–546, 1997.

[10] H. G. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 4th ed. London: Oxford University Press, 1979.

[11] J. A. Erho and J. I. Consul, "Precise evaluation of execution cost of sequence rotation by three-way-reversals," *International Journal of Applied Science Research*, vol. 2, pp. 99–104, 2019.

[12] C. A. Furia, "Rotation of sequences: Algorithms and proofs," Cornell University, USA, Tech. Rep., 2015.

[13] J. Bentley, "Code from programming pearls," in *Programming Pearls*, 2nd ed., P. Gordon, Ed. USA: ACM Press/Addison-Wesley Inc, 2000, ch. 'Column 2', pp. 140–143.

[14] B. Gastel and R. A. Day, *How to Write and Publish a Scientific Paper*, 8th ed. United States of America: GREENWOOD ABC-CLIO, LLC, 2016.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

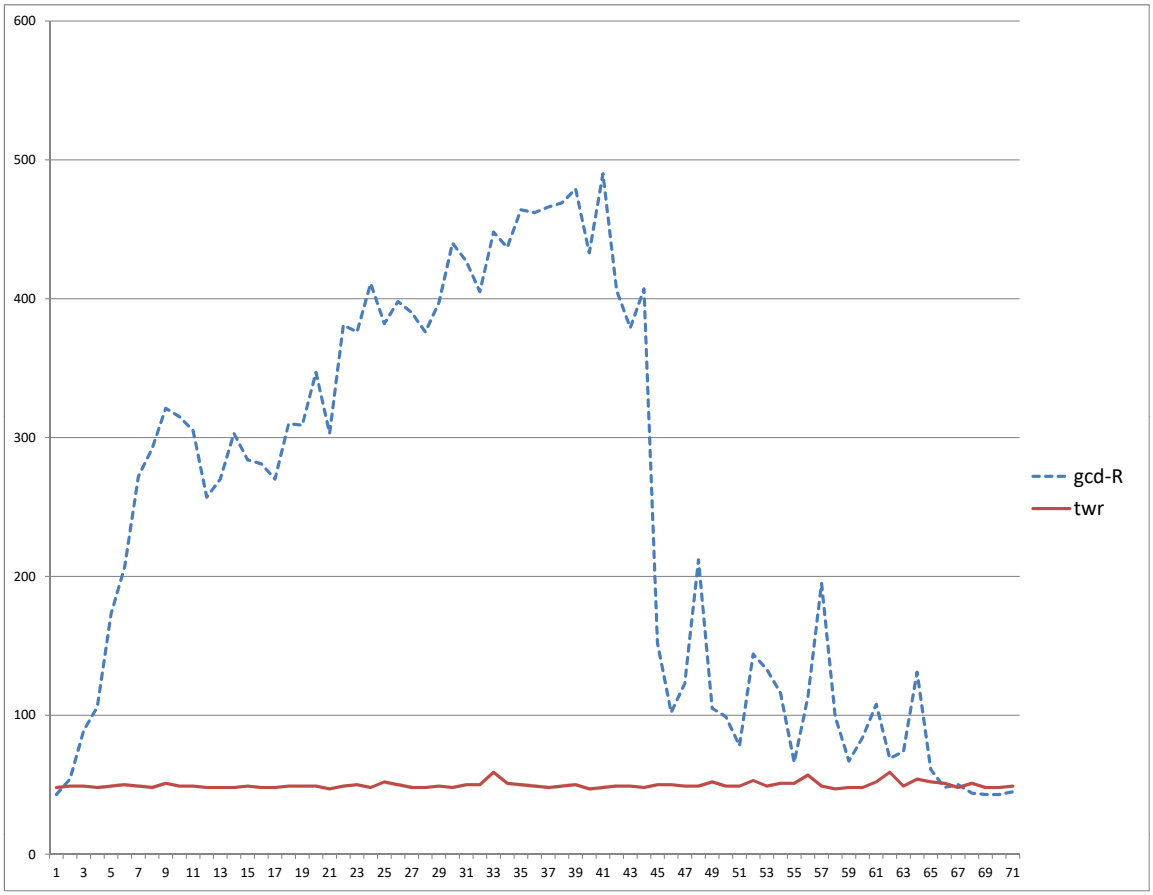

Fig. 1. gcd-R versus twr for LONG data type.

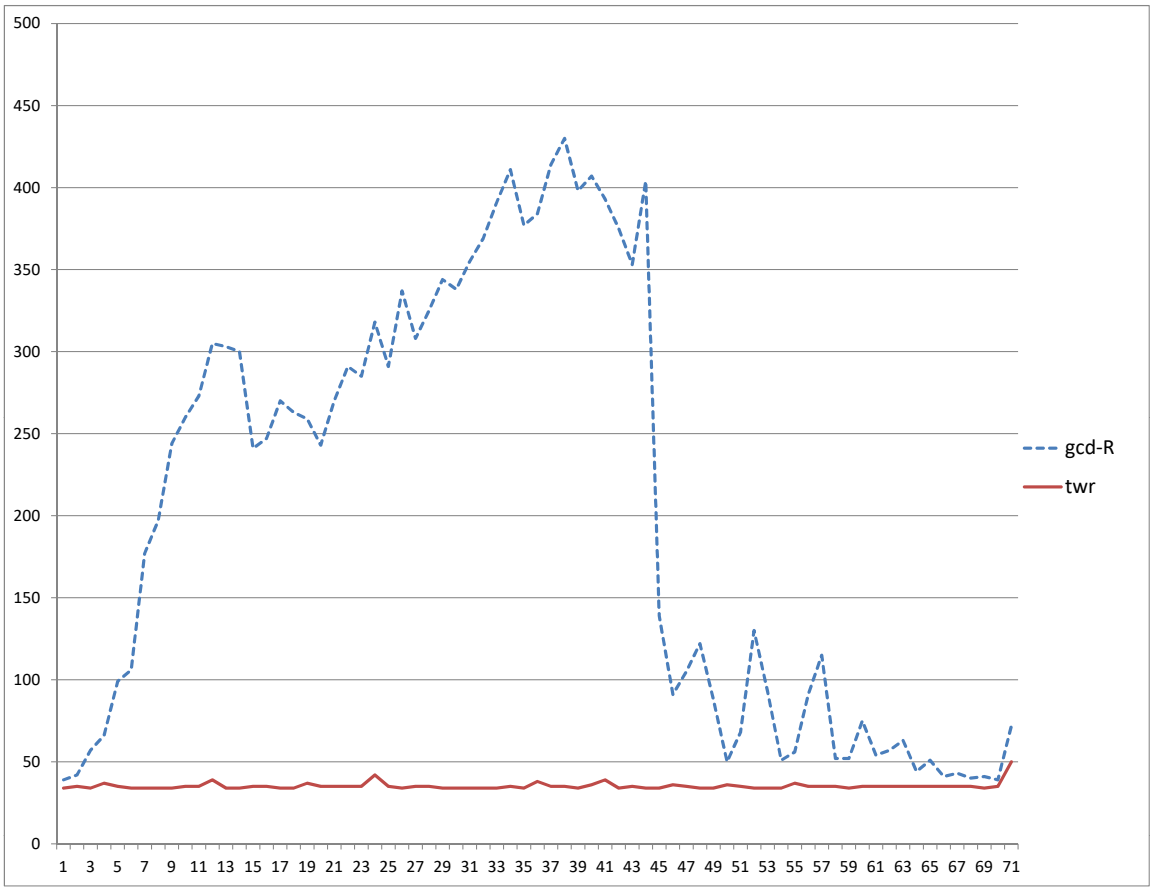

Fig. 2. gcd-R versus twr for INT data type.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

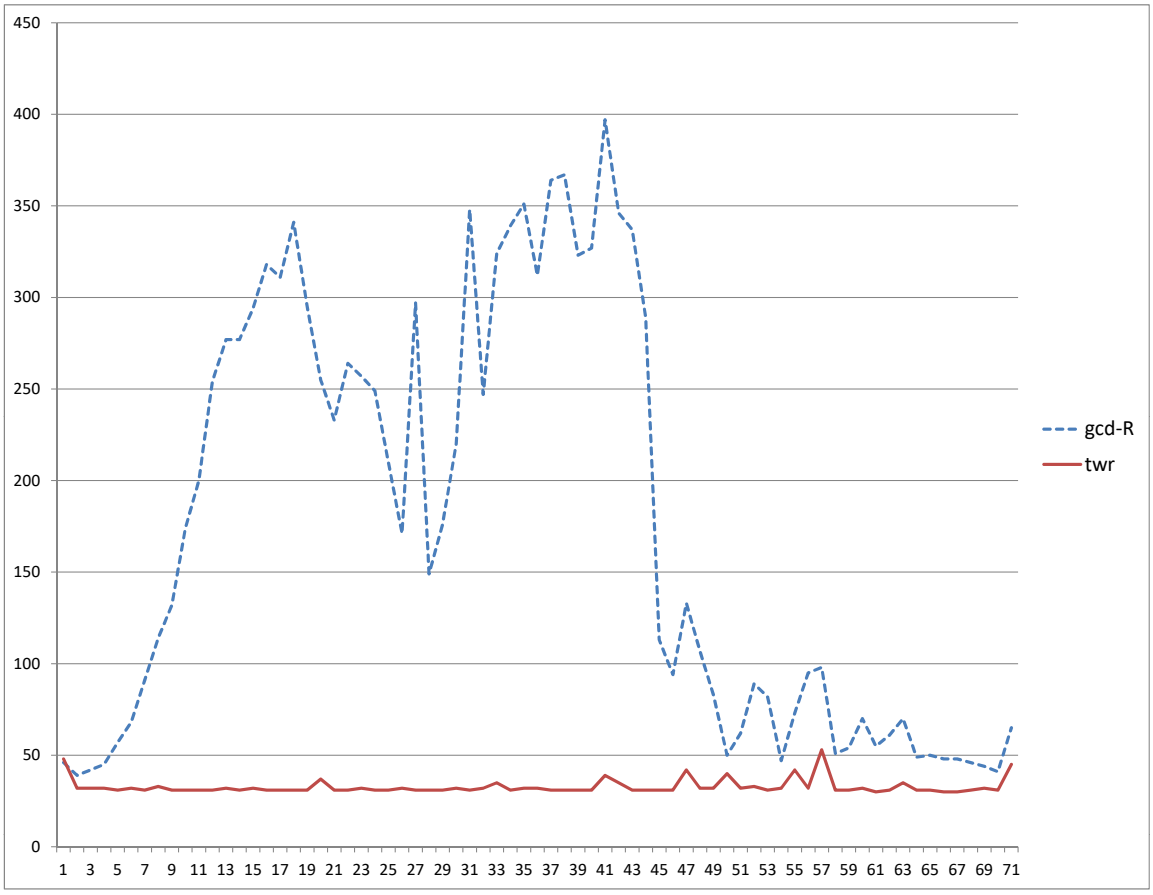

Fig. 3. gcd-R versus twr for SHORT data type.

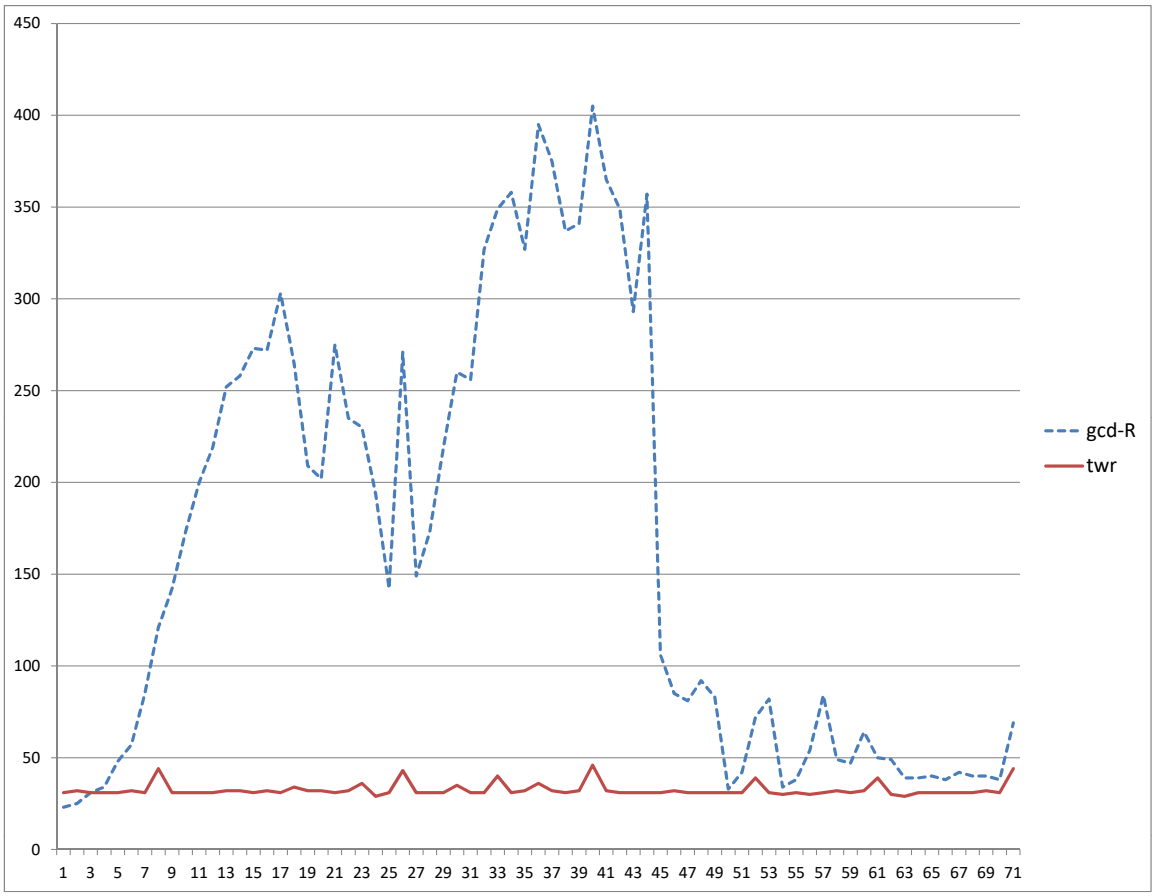

Fig. 4. gcd-R versus twr for CHAR data type.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)