Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:898
doi: 10.56294/sctconf2024898

Category: STEM (Science, Technology, Engineering and Mathematics)

ORIGINAL

Stack sort: a new approach with sorting network and a buffer

Clasificación por pila: un nuevo enfoque con red de clasificación y un búfer

S. Muthusundari[1], V. Devi[2], S. Sharath Kumar[1], D. Sudhish Reddy[1], Kannedari Uday Kiran[1], Pulimi Hanith Sai Kumar Reddy[1], Gosani Bhanu Sai Priya[1], Katragunta Yagna Priya[1]

[1]Department of CSE, R.M.D. Engineering College, Kavaraipettai. Tamil Nadu, India.
[2]Department of CSE, Hindustan College of Engineering and Technology. Coimbatore, India.

Cite as: Muthusundari S, Devi V, Sharath Kumar S, Sudhish Reddy D, Kiran KU, Kumar Reddy PHS, Sai Priya GB, Priya KY. Stack sort: a new approach with sorting network and a buffer. Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:898. https://doi.org/10.56294/sctconf2024898

Submitted: 05-02-2024 **Revised:** 24-04-2024 **Accepted:** 13-06-2024 **Published:** 14-06-2024

Editor: Dr. William Castillo-González

ABSTRACT

Knuth introduced the problem of stack sorting. Stack sorting was implemented by t stacks in series. In this paper, we propose a new dimension to stack sorting problem by introducing a stack with sorting network and a petty buffer. Instead of using t stacks in series, it helps to improve the performance by avoiding shuffles the stack. The basic idea behind in this paper is to perform a stack sorting with a single stack, and to achieve greater performance. In this novel approach, 2 bit buffer is compared to stack and insert the element into stack in order to avoid multiple stack. The result shows the time complexity of the proposed algorithm is O (n).

Keywords: Single Stack; Sorting Network; Buffer; Sort; Complexity.

RESUMEN

Knuth introdujo el problema de la ordenación de pilas. La ordenación de pilas se realizaba mediante pilas en serie. En este trabajo, proponemos una nueva dimensión al problema de ordenación de pilas introduciendo una pila con red de ordenación y un búfer pequeño. En lugar de utilizar t pilas en serie, ayuda a mejorar el rendimiento evitando barajar la pila. La idea básica que subyace en este trabajo es realizar una ordenación de pilas con una sola pila, y conseguir un mayor rendimiento. En este novedoso enfoque, el buffer de 2 bits se compara con la pila e inserta el elemento en la pila para evitar la pila múltiple. El resultado muestra que la complejidad temporal del algoritmo propuesto es O (n).

Palabras clave: Pila Única; Red de Ordenación; Buffer; Ordenación; Complejidad.

INTRODUCTION

Stack is defined as a linear type of data structure, plays a vital role in the field of computer science.[1] At the same time, stacks also model to access the items in a last-in first-out fashion. The two popular operations in a stack are push and pop. The top operation of stack is a one-end operation. By using push and pop operations, all Items are added to and removed from the top end. A stack is used to arrange a permutation or input, i1 = pr1, pr2,...,prn, in its most basic form.[2] Pushing the elements of pr1 into an initially empty stack causes the elements to be popped, creating an output permutation. The push and pop actions are often to determine the output permutation.[9]

In order to sort the input in the simplest stack sorting issue, one needs the output permutation to be 1, 2...

n. The values of the stack must always be stored in decreasing order as a requirement for this to work.

Enforcing this condition leads to a various methods to pop out from the stack. One of the approaches is "greedy algorithm" which performs pops operation of a stack element if pushing the next input element that has violated the increasing property of the stack. By another method, sorting permutations using t stacks in series when t > 1.[3] Greedy-increasing stacks in series are the name of a well-known and constrained model. Another approach can be used to determine upper boundaries on the variety of sortable permutations.

A pop stack is stack with the restriction that when the stack is popped, all entries in the stack must be output. Avis and Newborn introduced pop stacks and showed that if can be sorted with a pop stack, then it cannot contain three entries in the same relative order as 231 (for the same reason that 231 cannot be sorted by a stack), nor can it contain three entries in the same relative order as 312.[11]

Knuth has showed that there is an optimal algorithm, called Stacksort, which is able to sort every sortable permutation.[4] A stack is a push and pop sorting mechanism that operates on a last-in, first-out basis. According to Knuth, if π avoids the permutation 231, it can be sorted, meaning that by performing push and pop operations on the sequence π1,..., πn,[18] one may output the identity 1,..., n) precisely. It's simple to see that Stack sort has two essential characteristics:[5]

1. The pieces within the stack are kept in ascending order from top to bottom; the stack is growing.
2. The algorithm is incredibly greedy; as long as the stack keeps growing, it will always decide to execute S. The term "right greedy" in this context refers to the standard visual depiction, where the output permutation is on the left, the stack is in the middle, and the input permutation is on the right.

A number of particular scenarios have been taken into consideration because sorting with two stacks is an extremely challenging job in general. Two passes through a stack can sort some permutations, referred to as the "West-2-stack-sortable permutations" (see West). Alternatively said, these are the permutations that a right greedy algorithm can sort using two stacks connected in series.[6] Although West-2-stack-sortable permutations do not belong to a class, they can be described by means of generalized patterns.

Thus, we consider the two cases:

1. Sorting permutations with a pop stack followed immediately by a stack.
2. Sorting permutations with a pop stack, a queue, and then a stack.

Related work

Giulio[15] introduced a sorting machine with k +1 stack series. In their approach the first k series contains the elements from descending order and the last k series contains in the reverse order of the first k. they analyzed the series with left greedy approach and finally obtained an optimal sorting.

L. Pudwell[14] discussed the set of permutations through a pop up operation are sortable by passes method. From their study they derived finally and concluded that the set of two-pop-stack sortable permutations of length 2n+1 with exactly n ascents has an equal number of permutations with last block of size one as permutations with last block size greater than one.

Mingzhu[13] focused on the algorithm complexity and optimization of their proposed method improvement. Their experimental results are shown that the improved algorithm effective complexity management on both time and space, on their corresponding research. Their approach has improved the practicability.

Matúš[12] discussed the variant of problem with 2 stacks. They used the concept of minimum uncut approach for providing polynomial time reduction and shown O(√ log n)-approximation of their problem.

Rebecca[8] has examined the permutation that the right-greedy algorithm can sort on t stacks in series and the left-greedy algorithm can sort on t stacks in series. The number of permutations that can be sorted by t stacks in series is demonstrated to have a lower bound as well.

Stefan[19] described that a network with just two stacks that can exchange items has a lower bound of (n2). Take into account how the n items are arranged in the two stacks at midnight, or just before the first element is removed. This is a permutation of all the elements and is known as the midnight permutation of the process. Visualize the two stacks stuck together top to bottom. Surprisingly, the pair (π, σ) represents a sorting id on the two stacks network in a unique way. Keeping the stacks adhered together in a straight line from the start can help you visualize the procedure. Imagine an operating head going left to right over this arrangement.

Felix[7] have demonstrated that, even when the problem is restricted to complete networks, it is NP-hard to approximate the minimum number of shuffles within O (n1) for networks of k 4 stacks. They did this by connecting stack sorting to MIN k-PARTITION on circle graphs (for which we demonstrate a stronger in approximability result of independent).

König[10] have recently presented a mathematical model and a heuristic for an especially complex stacking problem, the goal of which is to minimize shuffles. They replicate stacking issues in great detail, including limiting stack heights that arise in the logistics of integrated steel manufacturing and in container terminal operation. The PSPACE-completeness of sortability determination is displayed. Lim[6] have introduced the basic concept of stack sorting to the reader.

The introduction of Stack sorting with restricted stacks was made.[16] That the set of σ-machines who's associated sortable permutations are not a class is counted by Catalan numbers. Furthermore, they thoroughly examined two specific σ-machines (namely =321 and =123), giving them a complete characterization and listing of their sortable permutations.

Colin[17] developed stack sorting that utilizes consecutive patterns to avoid them. Demonstrate that the maximum number of steps needed to send a permutation to a periodic point is 132. Numerous open problems and conjectures are included in the paper's conclusion.

Katalin[18] explored a generalized version of this algorithm where instead of avoiding a single decrease, the stack avoids a set of T permutations. They determined the T map for which set is bijective.

Proposed work

The new proposed method which consists of three basic components, they are:

1. Stack
2. Sorting network
3. Buffer

Stack: the function of the first component is to insert the elements into the stack in anon-increasing order, and to pop out the elements in increasing order. Once the element is popped out, again it cannot be entered into the stack.

Sorting network: the function of the second component is to compare the two elements and give the maximum or minimum as per our requirement.

Bit buffer: the function of the third component is to store the two elements into the stored array, and to evaluate the elements by min or max function.

In this paper, based on the three functions we perform any combination of the input that can be sorted by the single stack with sorting network and a buffer. Also, examples will be presented showing that the new approach is better than the existing methods by:

- No shuffles.
- Less iterations.
- Fast access.
- No additional stack.

The Proposed Architecture Model is Given below in figure 1.

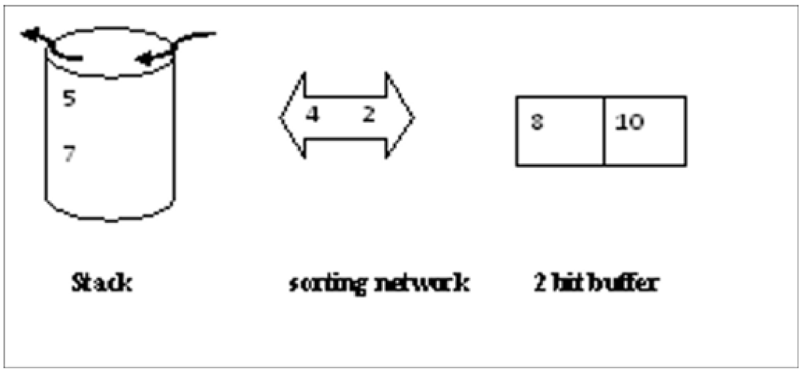


Figure 1. Proposed Architecture Model

The Working principle of the proposed model is given in the following figure 2.

1. The n permutation is given.
2. From the permutation first two bits are assigned into sorting network, and the next two bits are stored in the 2-bit buffer.
3. The sorting network compares the two elements and returns the maximum value.
4. The 2-bit buffer is used to store the next two elements into it and to return the maximum value by the max function.
5. Then the max (sorting network element, 2-bit buffer element) is assigned into the empty stack.
6. The next element from the permutation is considered to store in the empty portion again repeat the step 3 to step 6 to place all the elements into the stack.
7. While placing the elements into the stack, the stack rule is to adhere, if top[s] < new element then the element cannot be placed into the stack, it checks f or the next minimum of top[s] to be considered.
8. Otherwise, the new element is placed into the stack.

9. If top[s] < new element and there is no minimum of top[s], then pop out all the pushed elements from the stack, until the condition is true.

Case Study I that shows new element < top[s]

Permutation: 35421

Stack condition: new element < top[s], then the element is placed into the stack, otherwise the next minimum of top[s] is considered form the sorting network and the 2 bit buffer.

Sorting network: the first two elements 3 and 5 to be stored and returns the max value as 5.

2 bit buffer: the next 2 elements 4 and 2 to be stored in the buffer and returns the max value as 4 Now we take the max (5,4) = 5 is considered as new element.

The new element is compared with top[s]. In this case the stack is empty So the new element 5 is placed into the empty stack.

The next permutation is considered in the empty portion.

In sorting network 3 and 1 is stored, returns the max value as 3. In 2 bit buffer, there is no change, it returns the max value as 4.

Now we take the max (3,4) = 4, it shows 4<5, as per the stack condition, the new element 4 is placed into the stack.

There is no permutation, but still sorting network and buffer has elements to be placed into the stack. So, we compare the max (3,2) = 3, 3<4 hence 3 is placed into the stack.

Now 2 more data, it brings to sorting network 2 and 1, is compared and 2is placed then 1 is placed. Now there is a time to pop out the elements from the stack in the order as 1 2 3 4 5.

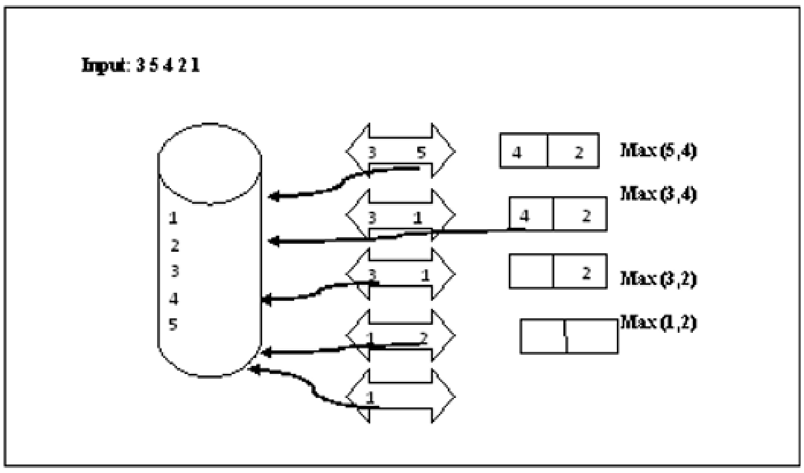


Figure 2. Implementation of case study I

Table 1. Stack sorting permutation 3 5 2 4 1

Input	Stack	Sorting network	Buffer	output
35241		3 5	2 4	
1	5	3 4	2	
	5	3 4	2 1	
	5 4	3	2 1	
	5 4	3 2	1	
	5 4 3	2 1		
	5 4 3 2	1		
	5 4 3 2 1			
				1
				1 2
				1 2 3
				1 2 3 4
				1 2 3 4 5

The stack sorting permutation is mentioned in table 1.

Case Study II for new element is not < top[s]

Permutation: 2 6 3 4 9 8 1 7 5

Stack condition: new element < top[s], then the element is placed into the stack, otherwise the next minimum of top[s] is considered form the sorting network and the 2 bit buffer.

Sorting network: the first two elements 2 and 6 to be stored and returns the max value as 6.

2 bit buffer: the next 2 elements 3 and 4 to be stored in the buffer and returns the max value as 4

Now we take the max (6,4) = 6 is considered as new element. In empty stack then new element 6 is placed. Now top[s] = 6.

In empty portion 6, the next element 9 is entered into sorting network, the max (2,9) = 9.

In 2 bit buffer, there is no change, it returns the max value as 4. Max (9,4) = 9. 9 is considered as new element. Now 9<6, in this case it fails, so 9 cannot be entered into the stack. It checks the next minimum of top[stack] as 6. The next minimum of 6 is 4. Hence the element 4 is placed into the stack.

In empty portion 4, the next element 8 is stored in 2 bit buffer. Then the max (2,9) = 9. In 2 bit buffer, the max (3,8) = 8. Now max (8,9) = 9. 9 is considered as new element. Now 9<4, in this case it fails, so 9 cannot be entered into the stack. It checks the next minimum of top[stack] as 4. The next minimum of 4 is 3. Hence the element 3 is placed into the stack.

In empty portion 3, the next element 1 is stored in 2 bit buffer. The max (1,8) = 8. In sorting network there is no change, the max (2,9) = 9. Now max (8,9) = 9. 9 is considered as new element. Now 9<3, in this case it fails, so 9 cannot be entered into the stack. It checks the next minimum of top[stack] as 3. The next minimum of 3 is 2. Hence the element 2 is placed into the stack.

In empty portion 2, the next element 7 is stored in sorting network. The max (7,9) = 9. In 2 bit buffer there is no change, the max (1,8) =8. Now max (8,9) = 9. 9 is considered as new element. Now 9<2, in this case it fails, so 9 cannot be entered into the stack. It checks the next minimum of top[stack] as 2. The next minimum of 2 is 1. Hence the element 1 is placed into the stack.

In empty portion 1, the next element 5 is stored in 2 bit buffer. The max (5,8) = 8. In sorting network there is no change, the max (7,9) =9. Now max (8,9) = 9. 9 is considered as new element. Now 9<1, in this case it fails, so 9 cannot be entered into the stack. It checks the next minimum of top[stack] as 1. The next minimum of 1 is not available in the sorting network and 2 bit buffer. Hence the elements are in the stack are popped out until it searches the next minimum element is available in the other two components based on the step 9 in the algorithm. Now the popped elements are 1 2 3 4. Now the next minimum of 6 is 5 considered here. So the element 5 is pushed in to the stack. Again, the condition fails to search for the next minimum element. Hence the elements 5 and 6 are popped out. Next the element 7 is pushed and popped out and 8 is pushed and pop out and 9 is pushed and pop out. Hence the order of the stack is 1 2 3 4 5 6 7 8 9.

The stack model is depicted in the figures 3, 4 and 5. The stack permutation is shown in table 2.

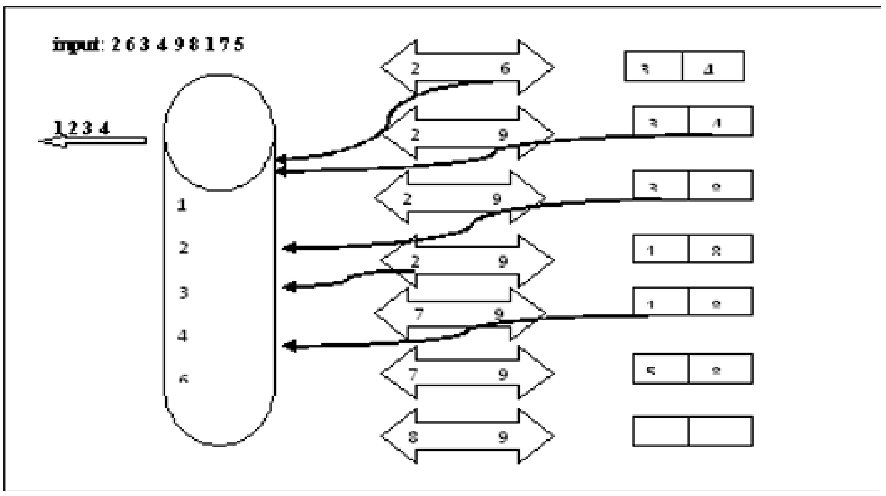


Figure 3. Implementation stages in case study II

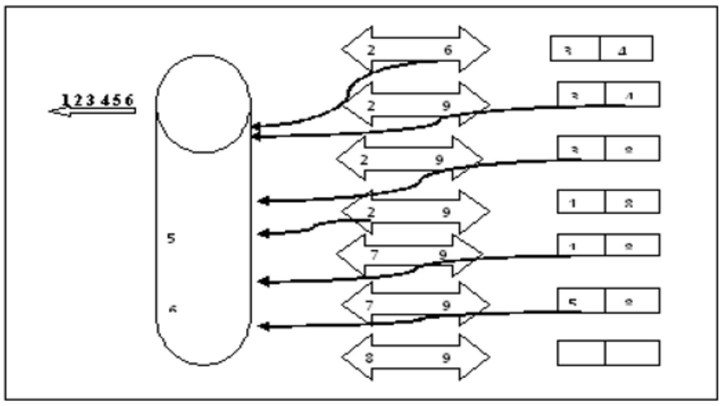


Figure 4. Implementation stages

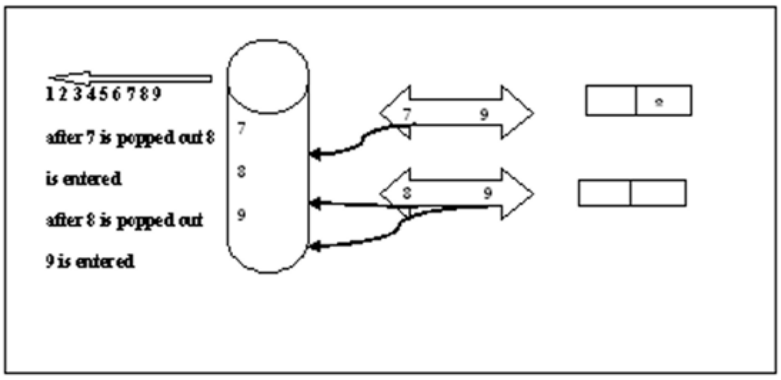


Figure 5. Implementation stages

Table 2. Stack sorting permutation 2 6 3 4 9 8 1 7 5

Input	Stack	Sorting network	Buffer	Output
263498175		2 6	3 4	
1	6	2 4	3 9	
	6 4	2 9	3 8	
	6 4 3	2 9	1 8	
	6 4 3 2	9 8	1 7	
	6 4 3 2 1	9 8	7 5	
	6 4 3 2 1	9 8	7 5	
	6 4 3 2 1	9 8	7 5	1
	6 4 3			1 2
	6 4			1 2 3
	6 5	9 8	7	1 2 3 4
	6	9 8	7	1 2 3 4 5
	7	9 8		1 2 3 4 5 6
	8	9		1 2 3 4 5 6 7
	9			1 2 3 4 5 6 7 8
				1 2 3 4 5 6 7 8 9

Performance analysis

We have measured the performance of the proposed method with the existing methods by the following characteristics.

1. It reduces the iterations performance by n-3.
2. The sorting network and buffer achieve fast access.

3. No additional stacks are required.
4. No shuffles also from stack to stack.

Complexity of the Algorithm

In this proposed algorithm, there are three stages the algorithm is defined.

- Stack Concept for insertion.
- Sorting network compares the maximum value.
- Stack Concept for deletion.

For implementing these stages 1 and 2 in stack concept, the time complexity for inserting and deleting the elements from the stack in the worst case it takes O(1) for inserting an element into the stack and deleting the elements it takes O(n). So let us take the stack stages the time complexity is O (n) for n permutation.

For finding the maximum value at each level for n permutations we have:

- At every individual step of the above said algorithm, at a time we need to compare two elements in the worst case Hence.
- Total no. of comparisons (in worst case) = 2*(n-1) = 2n – 2 (for all N permutations).
- Time Complexity for max function = O (n).
- Space complexity for max function in buffer = O (1).

Table 3. Complexity Analysis of the Proposed Algorithm

Approach	Complexity (Worst Case)
Stack Concept (Insertion)	O(1)
Max Function (Compares 2 at a time)	O(n)
Stack Concept (Deletion)	O(n)
Total no. of Comparisons	2*(n-1)

Hence the algorithm needs 2n-2 Comparisons and O (n) Complexity.

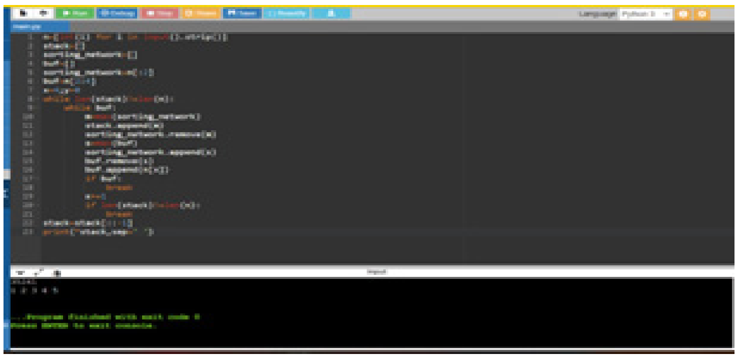
RESULTS AND DISCUSSION

The result of this proposed algorithm is compared with two stack and k stack series of existing methods in complexity analysis for the performance measurement.

Table 4. Result of Complexity and Comparison Analysis of the Proposed Algorithm

Approach	Complexity	Comparisons
Two Stack	O (√ log n)	n (n - 1) / 2
K Stack Series	O(n+k)	O(n^2)
Proposed Single Stack	O(n)	2n-2

Based on the result analysis is presented on the table 3 and 4. It shows the proposed algorithm is achieved a better performance than the existing methods. The Proposed algorithm has been implemented in python language. The output is shown in the figure 6.



```
n=[int(i) for i in input().strip()]
stack=[]
sorting_network=[]
buf=[]
sorting_network=n[:2]
buf=n[2:]
[illegible]
while len(stack)!=len(n):
    while buf:
        m=max(sorting_network)
        stack.append(m)
        sorting_network.remove(m)
        x=[illegible](buf)
        sorting_network.append(x)
        buf.remove(x)
        buf.append([illegible])
        if buf:
            break
        [illegible]
        if len(stack)!=len(n):
            break
stack=stack[::-1]
print("stack_seq=",")
```

```
[illegible]
1 2 3 4 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 6. Result of the proposed algorithm

CONCLUSIONS

In this paper, we have proven the performance of stack sorting problem with single stack and achieved a greater performance compared with existing method. In future the further refinements to be made on this proposed algorithm and to bring the performance to near O(log n) complexity.

BIBLIOGRAPHIC REFERENCES

1. Tarjan R. Sorting using networks of queues and stacks. Journal of the ACM (JACM). 1972 Apr 1; 19(2): 341-346.

2. Unger W. On the k-colouring of circle-graphs. In Annual Symposium on Theoretical Aspects of Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. 1988 Feb 11: 61-72.

3. Bóna M. A survey of stack-sorting disciplines. the electronic journal of combinatorics. 2002: A1. https://doi.org/10.37236/1693

4. Knuth DE. Fundamental Algorithms, vol. 1 of The Art of Computer Programming, section 1.2. Reading, Massachusetts: Addison-Wesley. 1973 Jan; 10: 10-19.

5. Zeilberger D. A proof of Julian West's conjecture that the number of two-stack sortable permutations of length n is 2 (3n)! / ((n+ 1) ! (2n+ 1)!). Discrete Mathematics. 1992 May 18; 102(1): 85-93.

6. LIM, C. H. Brief Introduction on Stack Sorting. https://www.math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/FINALAPP/Lim.pdf

7. König FG, Lübbecke ME. Sorting with complete networks of stacks. In International Symposium on Algorithms and Computation. Berlin, Heidelberg: Springer Berlin Heidelberg. 2008 Dec 15: 895-906.

8. Smith R. Comparing algorithms for sorting with t stacks in series. Annals of Combinatorics. 2004 May; 8(1): 113-121.

9. Atkinson MD, Murphy MM, Ruškuc N. Sorting with two ordered stacks in series. Theoretical Computer Science. 2002 Oct 23; 289(1): 205-223.

10. König FG, Lübbecke M, Möhring R, Schäfer G, Spenke I. Solutions to real-world instances of PSPACE-complete stacking. In European Symposium on Algorithms. Berlin, Heidelberg: Springer Berlin Heidelberg. 2007 Oct 8, pp. 729-740.

11. Marc P. The Bi-Stack Sorting Problem. Master's thesis, Department of Data Science and Knowledge Engineering, Maastricht University. 2019.

12. Mihalák M, Pont M. On Sorting with a Network of Two Stacks. In19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019) 2019.

13. Qian M, Wang X. Queue and stack sorting algorithm optimization and performance analysis. In AIP Conference Proceedings, AIP Publishing. 2018 Apr 18, 1955(1).

14. Pudwell L, Smith R. Two-stack-sorting with pop stacks. Australasian Journal of Combinatorics. 2019; 74(1): 179–195.

15. Cerbai G, Cioni L, Ferrari L. Stack Sorting with Increasing and Decreasing Stacks. The Electronic Journal of Combinatorics. 2020; 27(1): 1–17.

16. Cerbai G, Claesson A, Ferrari L. Stack sorting with restricted stacks. Journal of Combinatorial Theory, Series A. 2020 Jul 1; 173: 105230. https://doi.org/10.1016/j.jcta.2020.105230

17. Colin Defant, Kai Zheng. Stack-Sorting_with_Consecutive-Pattern-Avoiding_Stacks. 2020; August 2020. https://arxiv.org/pdf/2008.12297.pdf

18. Berlow K. Restricted stacks as functions. Discrete Mathematics. 2021 Nov 1; 344(11): 112571. https://doi.org/10.48550/arXiv.2008.01164

19. Felsner S, Pergel M. The Complexity of Sorting with Networks of Stacks and Queues. In: Halperin, D., Mehlhorn, K. (eds) Algorithms - ESA 2008. ESA 2008. Lecture Notes in Computer Science, vol 5193. Springer, Berlin, Heidelberg. 2008. https://doi.org/10.1007/978-3-540-87744-8_35.

FINANCING
No financing for the article.

CONFLICT OF INTEREST
The author declares that there is no conflict of interest in the work.

AUTHORSHIP CONTRIBUTION
Conceptualization: S. Muthusundari, V. Devi, S. Sharath Kumar, D. Sudhish Reddy, Kannedari Uday Kiran, Pulimi Hanith Sai Kumar Reddy, Gosani Bhanu Sai Priya, Katragunta Yagna Priya.

Data curation: S. Muthusundari, V. Devi, S. Sharath Kumar, D. Sudhish Reddy, Kannedari Uday Kiran, Pulimi Hanith Sai Kumar Reddy, Gosani Bhanu Sai Priya, Katragunta Yagna Priya.

Formal analysis: S. Muthusundari, V. Devi, S. Sharath Kumar, D. Sudhish Reddy, Kannedari Uday Kiran, Pulimi Hanith Sai Kumar Reddy, Gosani Bhanu Sai Priya, Katragunta Yagna Priya.

Methodology: S. Muthusundari, V. Devi, S. Sharath Kumar, D. Sudhish Reddy, Kannedari Uday Kiran, Pulimi Hanith Sai Kumar Reddy, Gosani Bhanu Sai Priya, Katragunta Yagna Priya.

Drafting - original draft: S. Muthusundari, V. Devi, S. Sharath Kumar, D. Sudhish Reddy, Kannedari Uday Kiran, Pulimi Hanith Sai Kumar Reddy, Gosani Bhanu Sai Priya, Katragunta Yagna Priya.

Writing - proofreading and editing: S. Muthusundari, V. Devi, S. Sharath Kumar, D. Sudhish Reddy, Kannedari Uday Kiran, Pulimi Hanith Sai Kumar Reddy, Gosani Bhanu Sai Priya, Katragunta Yagna Priya.