Sorting Magazines and Boxes

Gabriele Fici
University of Palermo, Italy

Manal Mohamed
King's College London, UK

Jakub Radoszewski
University of Warsaw, Poland

Abstract

It is a rainy Sunday. Agata has decided to sort the magazines on her shelf. Because the magazines are quite thin, she refuses to insert one between two others, preferring to move them only to the ends of the shelf. She has conceived a strategy for this but is unsure of its efficiency. Meanwhile, in the adjacent room, her three-year-old son, Szymon, has just finished his Montessori tower puzzle and is figuring out how to put it away. He has adopted a very intuitive approach to nesting the boxes, though he is not certain it will ultimately succeed.

Agata and Szymon are employing very primitive strategies. While many sorting algorithms are remarkably simple to explain and implement–specifically, the class of in-place sorting algorithms with $\mathcal{O}(n^2)$ worst-case and average-case running time and constant space requirements (e.g., Bubble Sort, Gnome Sort)–the strategies discussed here offer a unique perspective on "intuitive" sorting. Our contribution aims to enrich the field of simple sorting algorithms. Interestingly, determining the exact worst-case complexity of some of the proposed algorithms remains an open problem.

2012 ACM Subject Classification Theory of computation → Sorting and searching

Keywords and phrases Sorting algorithm, analysis of algorithms, intuitive sorting

Digital Object Identifier 10.4230/LIPIcs.FUN.2026.17

Funding *Gabriele Fici*: Supported by MIUR project PRIN 2022 APML – 20229BCXNW.

Acknowledgements We would like to thank all the participants of the Workshop *Frontiers of String Algorithms and Bioinformatics*, held at Birkbeck, London, UK, on Feb. 7, 2024, for useful discussions. The third author is indebted to his high school computer science teacher, Ms. Alina Gościniak, for introducing him to Naive Magazine Sort. He also thanks Jakub Pachocki for helpful discussions regarding Szymon's sorting algorithm.

1 Introduction

Agata has 10 magazines on her shelf. She wants to sort them according to her preferred order–for example, by issue number. Since magazines are thin, it is difficult to insert one between two others. Agata is therefore trying to devise a strategy that allows her to insert magazines only from the ends of the shelf.

Initially, Agata considers the following simple strategy:

"I pick the magazines from the shelf one by one and move them to a second shelf, inserting each magazine at the left end if it is smaller than the previous one, or at the right end otherwise."

Once all magazines are moved, the first shelf becomes empty, allowing her to start a second round with the same strategy–inverting the roles of the two shelves–if the magazines are not yet sorted. This strategy is presented in the pseudocode below. Queue M contains the magazines on the first shelf, and the double-ended queue (deque) M' represents the second shelf. We use standard queue operations: PopFront (removes and returns the first element), PushFront (inserts an element at the front), and PushBack (inserts an element at the back).

```
Naive Magazine Sort(M)
  repeat
    x ← PopFront(M)
    M' := {x}
    while M ≠ NIL do
      y := PopFront(M)
      if y < x then
        PushFront(M', y)
      else
        PushBack(M', y)
      x ← y
    M ← M'
  until M is sorted
```

In theory, there is no difference between theory and practice; in practice, there is[1]. When Agata considers how to implement her strategy *concretely*, she realizes that because the magazines are tall and thin, they cannot stand on their own. She would need an assistant to hold the magazines on one shelf while she manages the other.

The only other person home is her three-year-old son, Szymon, who is playing in the adjacent room. Agata must therefore improve her strategy. She realizes that because the shelves are in the corner of the room, the magazines are supported by the wall on the right end. Agata recalls reading that space can be used for multiple purposes simultaneously [4]. If she could eliminate the second shelf by reusing the space of the first shelf, she could support the magazines from the left end with one hand while using the other to pick and reinsert magazines.

[1] This quote is often attributed to baseball player Yogi Berra.

After some thought, Agata arrives at a new realization of her strategy:

"I scan all the magazines from left to right. Each time I find a magazine x that is smaller than the one immediately on its left, I mark x (by pulling it slightly towards me). Once I have finished scanning, I take each marked magazine in the order it was marked and reinsert them at the left end of the shelf."

This realization is equivalent to the first but significantly more convenient to perform manually. The following pseudocode illustrates this strategy using a single shelf M, implemented as a doubly-linked list with operations: Front (returns the first element), Next (returns the subsequent element), and MoveToFront (removes a given element from its current position and reinserts it at the front of the list). We utilize a queue *Marked* to store pointers to the marked elements, allowing for $\mathcal{O}(1)$ reinsertion.

```
Magazine Sort With Marking(M)
  Marked ← ∅
  repeat
    x ← Front(M)
    while Next(M, x) ≠ NIL do
      if Next(M, x) < x then
        PushBack(Marked, Next(M, x))
      x ← Next(M, x)
    while Marked ≠ ∅ do
      x ← PopFront(Marked)
      MoveToFront(M, x)
  until M is sorted
```

▶ **Example 1.** For convenience, let's denote the magazines by their rank in the sorted order:

Round 0: $(4, 3, 1, 5, 2, 6, 10, 8, 9, 7)$.

Agata marks magazines $3, 1, 2, 8$ and 7, as these are the elements preceded by a larger one. She then picks the marked elements one by one, in the order they were encountered from left to right, and reinserts them at the left end of the shelf. The resulting order is:

Round 1: $(7, 8, 2, 1, 3, 4, 5, 6, 10, 9)$.

After ten rounds, Agata starts to wonder if this procedure will ever terminate. Is she simply repeating the same sequence of moves? However, the last three magazines are now in the correct order, which sounds promising... She decides to persevere, and finally, at the end of the 20th round, the magazines are completely sorted!

▶ **Example 2** (Example 1 continued)**.** Agata will finish after 20 rounds. The sequence of rounds demonstrates the gradual movement of smaller elements toward the front and the emergence of sorted suffixes.

Round 1:	(7, 8, 2, 1, 3, 4, 5, 6, 10, 9)	Round 11:	(3, 4, 1, 2, 7, 5, 6, 8, 9, 10)
Round 2:	(9, 1, 2, 7, 8, 3, 4, 5, 6, 10)	Round 12:	(5, 1, 3, 4, 2, 7, 6, 8, 9, 10)
Round 3:	(3, 1, 9, 2, 7, 8, 4, 5, 6, 10)	Round 13:	(6, 2, 1, 5, 3, 4, 7, 8, 9, 10)
Round 4:	(4, 2, 1, 3, 9, 7, 8, 5, 6, 10)	Round 14:	(3, 1, 2, 6, 5, 4, 7, 8, 9, 10)
Round 5:	(5, 7, 1, 2, 4, 3, 9, 8, 6, 10)	Round 15:	(4, 5, 1, 3, 2, 6, 7, 8, 9, 10)
Round 6:	(6, 8, 3, 1, 5, 7, 2, 4, 9, 10)	Round 16:	(2, 1, 4, 5, 3, 6, 7, 8, 9, 10)
Round 7:	(2, 1, 3, 6, 8, 5, 7, 4, 9, 10)	Round 17:	(3, 1, 2, 4, 5, 6, 7, 8, 9, 10)
Round 8:	(4, 5, 1, 2, 3, 6, 8, 7, 9, 10)	Round 18:	(1, 3, 2, 4, 5, 6, 7, 8, 9, 10)
Round 9:	(7, 1, 4, 5, 2, 3, 6, 8, 9, 10)	Round 19:	(2, 1, 3, 4, 5, 6, 7, 8, 9, 10)
Round 10:	(2, 1, 7, 4, 5, 3, 6, 8, 9, 10)	Round 20:	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10).

In each round, Agata first scans all the magazines to decide which ones to mark. In our example, Agata marked and subsequently moved 52 magazines across all rounds. Assuming it takes her 1 second to check if a magazine should be marked, 1 second to mark it, and 2 seconds to move it, the total time spent was $20 \cdot 9 + 52 \cdot 3 = 336$ seconds–less than 6 minutes to sort her 10 magazines.

Agata wonders whether she was simply lucky this time, or if her strategy is guaranteed to work... Then she pauses: "Is this marking phase really necessary after all?" She devises a final improvement:

"I scan the magazines from left to right. Whenever I find a magazine x smaller than the one on its left, I immediately pick x and reinsert it at the left end."

This refined strategy is illustrated in the pseudocode below. In this version, the move is performed immediately, eliminating the need for a marking queue. Note that when an element y is moved to the front, the next element to be checked is compared against the same x that was previously to the left of y.

```
MAGAZINE SORT NO MARKING(M)
  repeat
    x ← FRONT(M)
    while NEXT(M, x) ≠ NIL do
      y ← NEXT(M, x)
      if y < x then
        MOVETOFRONT(M, y)
      else
        x ← NEXT(M, x)
  until M is sorted
```

"Well, this actually seems faster!" she thinks. Indeed, this time she finishes after only 6 rounds.

▶ **Example 3** (Example 1 continued)**.** We start with the same initial arrangement:

Round 0: $(4, 3, 1, 5, 2, 6, 10, 8, 9, 7)$.

In the first round, Agata picks 3, as it is smaller than 4 (the element immediately to its left), and reinserts it at the beginning of the shelf. She then picks 1, as it is smaller than 4 – which is *now* the element immediately to the left of 1 after 3 was moved. This process continues through the shelf. The final elements moved to the beginning of the shelf are 8 (as 10 is on its left), then 9 (as 10 is still on its left), and finally 7. This concludes the first round.

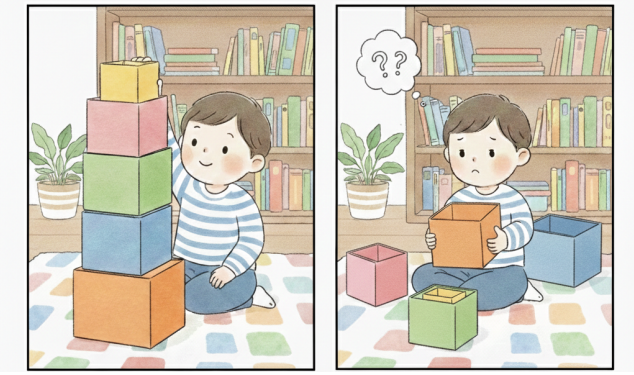

Figure 1 A Montessori tower consisting of cubes of varying sizes. Each cube has one missing face, allowing smaller cubes to be nested or stacked.

Agata completes the sorting in 6 rounds, as shown below:

Round 1: $(7, 9, 8, 2, 1, 3, 4, 5, 6, 10)$
Round 2: $(6, 5, 4, 3, 1, 2, 8, 7, 9, 10)$
Round 3: $(7, 2, 1, 3, 4, 5, 6, 8, 9, 10)$
Round 4: $(6, 5, 4, 3, 1, 2, 7, 8, 9, 10)$
Round 5: $(2, 1, 3, 4, 5, 6, 7, 8, 9, 10)$
Round 6: $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$.

In total, Agata moved 31 magazines over 6 rounds. Assuming she needs 1 second to determine if a magazine should be moved and 2 seconds to move each magazine, the total time required was $6 \cdot 9 + 31 \cdot 2 = 116$ seconds–less than 2 minutes to sort her 10 magazines.

At this point, we are left with several questions: Do all of Agata's strategies always converge to the sorted list? And if so, what is the number of rounds or steps in the worst and average cases?

In the meantime, in the adjacent room, Agata's three-year-old son, Szymon, is playing with his favorite toy: a Montessori tower (see Figure 1).

The toy consists of ten cubic boxes of different sizes. After Szymon successfully builds the tower by stacking all the cubes in decreasing order of size, he enjoys toppling. Now the boxes lie scattered across the carpet. Szymon knows that each box has one missing face, which must serve as the bottom; therefore, to put the toy away, he must nest the boxes one inside another in the correct order.

Szymon employs a simple, intuitive algorithm: First, he selects a box he assumes will be the outermost. He then picks up another box with his right hand and attempts to place it inside the first. He continues this process as long as each subsequently selected box fits within the one immediately preceding it. If the box does not fit, Szymon repeatedly removes the innermost box with his left hand until the box currently in his right hand fits inside. The boxes removed from the partial stack are returned to the carpet in a disordered fashion. The specific order in which Szymon selects the remaining boxes is not immediately evident.

The following pseudocode illustrates the strategy. The partial order of boxes is implemented as a stack S, with the largest box at the bottom and the smallest at the top. The stack provides standard methods: Push (adds an element to the top), Pop (removes the top

element), and Top (returns the top element). The collection M of boxes remaining on the carpet supports the operations: Insert (returns a box to the collection) and RandomPick (selects a box based on Szymon's undisclosed selection criterion).

```
Montessori Tower Random Sort(M)
  S ← ∅                                  ▷ Initialize stack for the partially ordered set
  while M ≠ ∅ do
    x ← RandomPick(M)
    while S ≠ ∅ and Top(S) < x do
      y ← Pop(S)
      Insert(M, y)
    Push(S, x)
  return S
```

Szymon is always able to order all the boxes within a few minutes, even though it is not immediately obvious to his parents that his algorithm is guaranteed to terminate. Moreover, the worst-case complexity of his approach may seem concerning. Was he simply lucky with the sequence of boxes he chose, or did he have some underlying intuition?

In Section 2, we analyze the complexity of Agata's magazine sorting algorithms, while in Section 4 we examine the complexity of Szymon's Montessori sorting algorithm. Finally, in Section 3, we discuss further potential improvements and experimental results.

2 Analysis of magazine sorting algorithms

Let us first analyze Agata's algorithms. In full generality, let n be the number of magazines to be sorted. For simplicity, we assume magazines are identified with their ranks $1, \ldots, n$, though the algorithms only require a total strict order $<$ and do not assume a specific permutation of integers. We further assume all magazines are distinct.

As we have already mentioned, Naive Magazine Sort and Magazine Sort With Marking are essentially equivalent. Specifically, they perform the same number of rounds. Each round can be implemented in time linear in the number of magazines using appropriate pointer data structures: a queue and a double-ended queue for Naive Magazine Sort, and a list and a marking-queue for Magazine Sort With Marking. Thus, the crucial parameter in both algorithms is the number of rounds.

Let us analyze Magazine Sort With Marking. The algorithm consists of one or more rounds where magazines are scanned from left to right. In any given round, let i be the largest element currently in a position smaller than i. It follows that element $i + k$ is in position $i + k$ for every $k > 0$, and it will remain there after each subsequent round. In the next round, i will be moved to position $i + j$ for some $j > 0$, as there is at least one element to the right of i that is smaller than its predecessor and will thus be moved to the front. This demonstrates that the algorithm converges in at most $n(n+1)/2$ rounds or $\mathcal{O}(n^3)$ total steps.

An almost-worst-case instance appears to be the permutation $(n, 1, 2, \ldots, n-1)$. For this permutation, it takes exactly $n - 1$ rounds for n to reach the final position. After the i-th round, element n is at position $i + 1$, followed by $i + 1, \ldots, n - 1$:

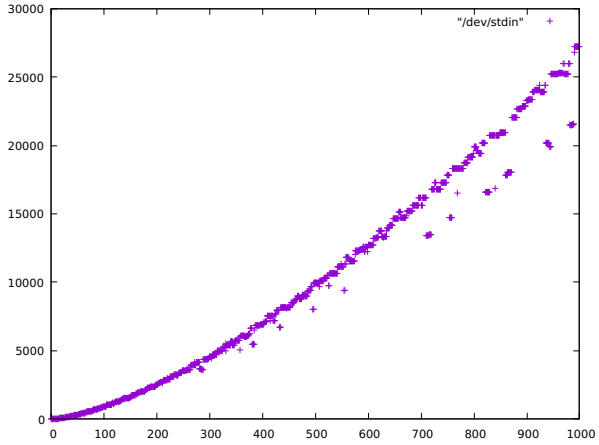

Figure 2 The graph contains a point $(n, f(n))$ for every $n = 1, \ldots, 1000$ such that Magazine Sort With Marking for permutation $(n, 1, 2, \ldots, n-1)$ takes $f(n)$ rounds.

Round 0: $(n, 1, 2, 3, 4, 5, \ldots, n-1)$
Round 1: $(1, n, 2, 3, 4, 5, \ldots, n-1)$
Round 2: $(2, 1, n, 3, 4, 5, \ldots, n-1)$
Round 3: $(3, 1, 2, n, 4, 5, \ldots, n-1)$
Round 4: $(4, 1, 3, 2, n, 5, \ldots, n-1)$
$\ldots$

While the permutation $(n, 1, 2, \ldots, n-1)$ seems to require $\omega(n)$ rounds, we have not yet proven this theoretically. Figure 2 contains a graph showing the number of steps the algorithm performed on permutations of this type.

▶ Remark 4. We have experimentally verified that the permutation $(n, 1, 2, \ldots, n-1)$ is a worst-case instance for $n = 1, 2, \ldots, 8$ and $n = 10$. For $n = 9$, this permutation requires 24 rounds, whereas there exist other permutations, such as $(3, 9, 4, 5, 1, 2, 6, 7, 8)$, which require 26 rounds. Similarly, it is not the worst-case instance for $n = 11$ or $n = 12$.

Determining the exact complexity of Magazine Sort With Marking remains an open problem.

We now analyze the Magazine Sort No Marking algorithm. This variant performs at most $n-1$ rounds. To see this, let i be the largest element not in the correct sorted position i at the start of a round; specifically, suppose it is in position $i-j$ for some $j > 0$. By the maximality of i, all elements to the right of i up to position i (i.e., those in position $i-j+1$ through i) must be smaller than i. Consequently, these elements will be moved to the front by the end of the current round. This guarantees that i will occupy its correct position i by the next round.

Thus, the algorithm requires at most $n-1$ rounds and is therefore worst-case quadratic ($\mathcal{O}(n^2)$ in the total number of element moves).

A worst-case instance for odd $n = 2k-1$ is the permutation

$(2k-1, 2k-3, \ldots, 3, 1, 2, 4, \ldots, 2k-2)$.

This instance is constructed in such a way that before the i-th round, for $i = 1, \ldots, n-1$, the element $n+1-i$ is at the beginning, while all elements greater than $n+1-i$ (the $i-1$ largest elements) are already correctly placed at the end. This forces the algorithm to perform exactly $n-1$ rounds. For example:

Round 0: $(2k-1, 2k-3, 2k-5, \ldots, 3, 1 | 2, 4, \ldots, 2k-4, 2k-2)$
Round 1: $(2k-2, 2k-4, \ldots, 4, 2 | 1, 3, \ldots, 2k-5, 2k-3 | 2k-1)$
Round 2: $(2k-3, 2k-5, \ldots, 3, 1 | 2, 4, \ldots, 2k-4 | 2k-2, 2k-1)$
Round 3: $(2k-4, \ldots, 4, 2 | 1, 3, \ldots, 2k-5 | 2k-3, 2k-2, 2k-1)$
$\ldots$

A similar worst-case instance for even $n = 2k$ is $(2k, 2k-2, \ldots, 4, 2, 1, 3, \ldots, 2k-1)$.

Comparison with other classic sorting algorithms. Magazine Sort No Marking shares some characteristics with Bubble Sort, but they behave differently. In Bubble Sort, if the largest element is at the beginning, it moves to the end in one round while the relative order of the remaining elements is preserved. In Magazine Sort No Marking, the maximal element also reaches the end in one round, but the relative order of the other elements is *reversed*. The performance can diverge significantly: for the reverse permutation $(n, n-1, \ldots, 1)$, Bubble Sort is quadratic, whereas Magazine Sort No Marking requires only one round. Furthermore, Bubble Sort is physically impractical for sorting magazines because it requires swaps or insertions in the middle of the sequence.

Selection Sort is physically compatible with the magazine model. However, Selection Sort always performs $\Theta(n^2)$ comparisons regardless of the input, while Magazine Sort No Marking can be faster on specific nearly-sorted or reversed inputs.

Insertion Sort and Gnome Sort typically run in $\mathcal{O}(n+I)$ time, where I is the number of inversions. Like Bubble Sort, they perform $\Theta(n^2)$ steps for the reversed sequence $(n, n-1, \ldots, 1)$–an "easy" case for our algorithm–and they do not fit the magazine model due to the requirement for the middle-sequence insertion.

3 Optimizing magazine sorting algorithms

The implementation of the Magazine Sort With Marking algorithm can be improved so that in each round (except for the first), the total number of operations is proportional only to the number of moved elements. The core logic remains unchanged; intuitively, during the moving phase of the current round, we can simultaneously perform the marking for the next round.

Consider a round $i \geq 1$, and let x be an element that is marked by the original Magazine Sort With Marking algorithm. This implies that its predecessor, $y = \textsc{Prev}(M, x)$, satisfies $y > x$. Now consider round $i-1$. Let y' denote $\textsc{Prev}(M, x)$ in round $i-1$. We have $y' \neq y$, as otherwise x would have been marked and moved in round $i-1$. This implies that either y' or x was moved in round $i-1$.

In the optimized algorithm, whenever an element u is removed from the list, we check if the subsequent element v should be marked for the next round by comparing it with the predecessor of u. This corresponds to marking $v = x$ when $u = y'$ is moved. Furthermore, whenever an element u is prepended to the list, we check if the element v that was previously at the front should be marked for the next round. This corresponds to marking $v = x$ when $u = y$ is moved to the front.

In the first round of Optimized Magazine Sort With Marking, the marking phase is exactly the same as in Magazine Sort With Marking. The list *Marked* stores all the marked elements for the current round, and the list *Marked*′ stores all the marked elements for the next round. We then iterate through all elements of the list *Marked* and move them to the front one by one. Whenever a marked element is removed from the list M, we check if its successor in the list should be marked for the next phase; if so, we add it to *Marked*′. Additionally, when the marked element is pushed to the front of the list M, we check if the previously first element of the list should be marked for the next round. In the end, *Marked*′ becomes the new list *Marked*.

```
Optimized Magazine Sort With Marking(M)
  Initialization:
  Marked ← Marked′ ← ∅
  x ← Front(M)
  while Next(M, x) ≠ NIL do
    if Next(M, x) < x then
      PushBack(Marked, Next(M, x))
    x ← Next(M, x)
  Main algorithm:
  repeat
    while Marked ≠ NIL do
      x ← PopFront(Marked)
      if (Next(M, x) < Prev(M, x)) and (Next(M, x) > x) then
        PushBack(Marked′, Next(M, x))
      if Front(M) < x then
        PushBack(Marked′, Front(M))
      MoveToFront(M, x)
    Marked ← Marked′
    Marked′ ← ∅
  until Marked = NIL
```

▶ **Example 5.** During initialization: Agata marks magazines $3, 1, 2, 8$ and 7.

Round 0: $(4, \underline{3}, \underline{1}, 5, \underline{2}, 6, 10, \underline{8}, 9, \underline{7})$; marked elements are underlined.

In the first round, Agata picks the marked elements from left to right. Upon removing an element, she checks the new neighbors; if the successor is smaller than the predecessor, she marks it. Thus, she marks 9 when 8 is removed. Similarly, when reinserting at the left, if she places a magazine to the left of a smaller one, she marks the smaller one. Thus, 1 is marked when 2 is reinserted, and 2 is marked when 8 is reinserted. By the end of Round 1, the shelf is:

Round 1: $(7, 8, \underline{2}, \underline{1}, 3, 4, 5, 6, 10, \underline{9})$.

In Round 1, when element 2 is moved, element 1 will *not* be marked for Round 2, as it is already marked for Round 1 and will be moved within the same round. Next, element 3 will be marked for Round 2. Actually, element 1 will be marked at its new position at the same moment element 9 is moved to the front. Overall, the shelf is ready for the second round and looks as follows:

Round 2: $(9, \underline{1}, 2, 7, 8, \underline{3}, 4, 5, 6, 10)$.

Agata continues removing marked elements, marking more as she progresses. She ends up marking and removing 52 magazines in total. However, she no longer needs to scan the entire shelf in each round.

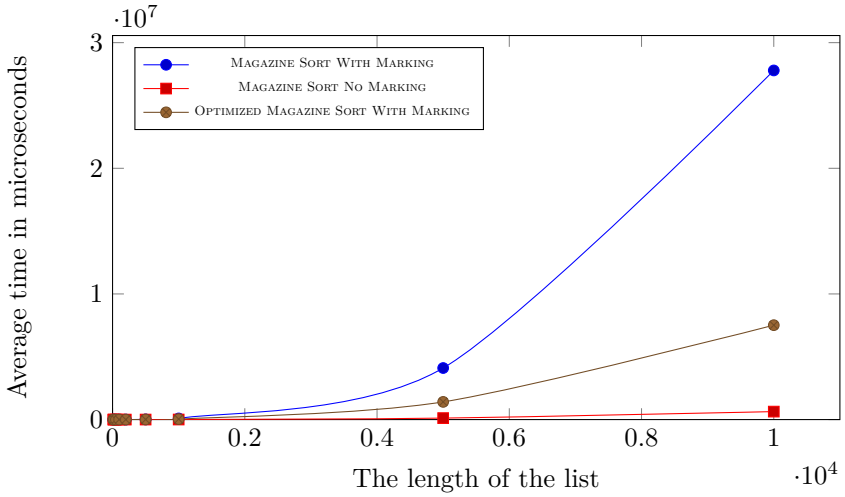

Figure 3 The graph illustrates the average running time in microseconds over 5 randomly generated lists. Optimized Magazine Sort With Marking is faster than the original version, though Magazine Sort No Marking remains superior.

In Optimized Magazine Sort With Marking, there is no strict correspondence between the number of rounds and the total number of steps. The algorithm performs the same number of rounds as its predecessor, but each round is faster. This is confirmed by experimental analysis in Figure 3. The upper bound of $\mathcal{O}(n^3)$ steps still applies. As a comparison-based algorithm, the classic $\Omega(n \log n)$ lower bound applies. The exact complexity, however, remains an open problem.

We can similarly optimize Magazine Sort No Marking by ensuring its complexity depends only on the number of magazines effectively moved. This requires reintroducing a focused marking strategy to track elements affected by displacements.

In the first round of Optimized Magazine Sort No Marking, the initialization is identical to the previous optimized algorithm. The list *Marked* stores current marked elements, and *Marked*$'$ stores the marked elements for the next round. When a marked element x is processed, it is smaller than its predecessor $y = \textsc{Prev}(M, x)$. Therefore, x and all elements to its right that are smaller than y are moved to the front one by one. When an element is pushed to the front, it is removed from *Marked* (if present), and we check if the previously first element of the list should be marked for the next round.

Optimized Magazine Sort No Marking(M)
  Initialization: As in Optimized Magazine Sort With Marking
  Main algorithm:
  repeat
    while *Marked* $\neq$ *NIL* **do**
      $x \leftarrow \textsc{PopFront}(\textit{Marked})$
      $y \leftarrow \textsc{Prev}(M, x)$
      while $x < y$ **do**
        if $\textsc{Front}(M) < x$ **then**

```
            PushBack(Marked′, Front(M))
        MoveToFront(M, x)
        x ← Next(M, y)
        if x = Front(Marked) then
            PopFront(Marked)
    Marked ← Marked′
    Marked′ ← ∅
until Marked′ = NIL
```

▶ **Example 6.** Beginning with the initial arrangement of magazines from Example 5:

Round 0: $(4, \underline{3}, \underline{1}, 5, \underline{2}, 6, 10, \underline{8}, 9, \underline{7})$.

In the next round, Agata picks 3 (the first marked element) and reinserts it at the beginning, followed by 1. However, when Agata picks 2, before reinserting it, she marks 1 (the current leftmost element) since $1 < 2$. Similarly, she marks 2 before reinserting 8, and marks 8 before reinserting 9. By the end of the first round:

Round 1: $(7, 9, \underline{8}, \underline{2}, \underline{1}, 3, 4, 5, 6, 10)$

Processing only marked elements, Agata finishes after 5 more rounds:

Round 2: $(6, 5, \underline{4}, \underline{3}, \underline{1}, 2, 8, 7, 9, 10)$

Round 3: $(7, \underline{2}, \underline{1}, 3, 4, 5, 6, 8, 9, 10)$

Round 4: $(6, \underline{5}, \underline{4}, \underline{3}, \underline{1}, 2, 7, 8, 9, 10)$

Round 5: $(2, \underline{1}, 3, 4, 5, 6, 7, 8, 9, 10)$

Round 6: $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$.

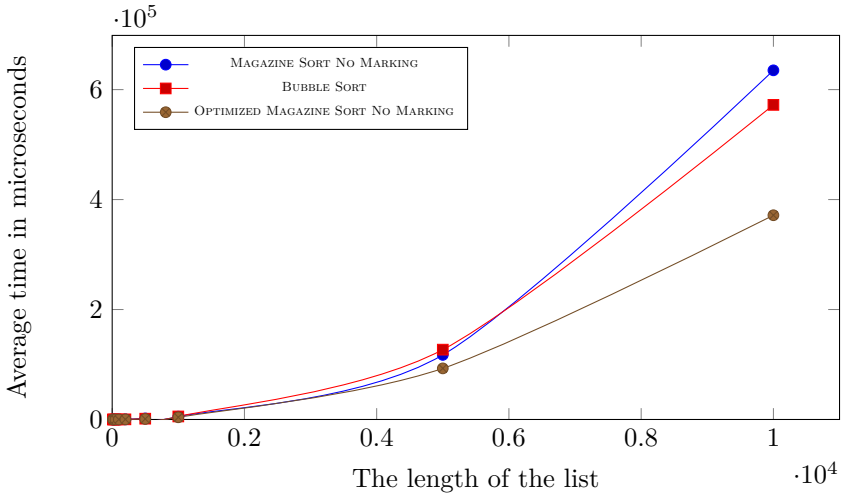

Figure 4 Experiments on randomly generated lists show that the "marking while moving" strategy in Optimized Magazine Sort No Marking provides the best performance among Agata's algorithms.

The optimized Magazine Sort No Marking maintains a worst-case complexity of $\Theta(n^2)$. However, as shown in Figure 4, it performs significantly better than the basic version and Bubble Sort on random permutations.

A summary of the complexities of magazine sorting algorithms can be found in Table 1.

Table 1 Worst-case time complexity bounds for the magazines sorting algorithms in the comparison model.

Algorithm	Lower Bound	Upper Bound
Magazine Sort With Marking	$\Omega(n^2)$	$\mathcal{O}(n^3)$
Optimized Magazine Sort With Marking	$\Omega(n \log n)$	$\mathcal{O}(n^3)$
Magazine Sort No Marking	$\Omega(n^2)$	$\mathcal{O}(n^2)$
Optimized Magazine Sort No Marking		

4 Analysis of Montessori sorting algorithm

Let us now analyze Szymon's algorithm in a general setting with n boxes, of sizes 1 (smallest) to n (largest).

As established, the "partial order" of boxes stored in Montessori Tower Random Sort functions as a stack. At each step, a box is selected from the carpet; it is pushed onto the stack if its size is smaller than the box currently on top. Otherwise, boxes are popped from the stack one by one until the top box is larger than the box to be inserted.

First, we prove that the algorithm always terminates. The set of boxes stored on the stack can be represented as a binary sequence of length n where the i-th bit is 1 if box i is on the stack and 0 otherwise. Suppose box j is selected from the carpet. The algorithm's operations consist of zeroing all bits at positions $1, \ldots, j-1$ (the pops) and setting the j-th bit to 1 (the push). If the sequence is viewed as a binary counter, where the n-bit is the most significant digit, then every step in the algorithm strictly increases the counter's value. The counter starts at 0 (an empty stack) and reaches a maximum at $2^n - 1$ (all boxes correctly nested). Since the counter strictly increases with each selection, the algorithm must terminate.

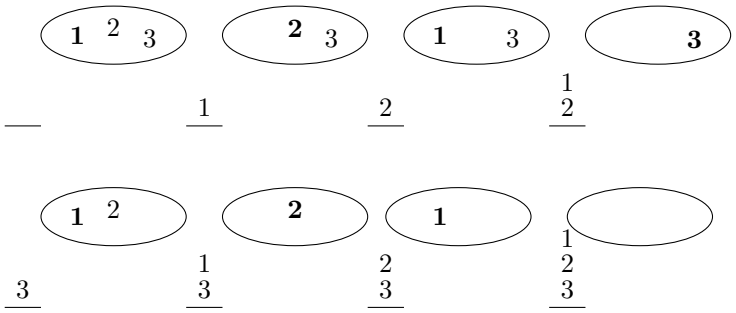

Figure 5 The worst-case execution of Montessori Tower Random Sort for $n = 3$ boxes. Encircled numbers represent free boxes on the carpet, and the stack represents the partial order. The bold number indicates the box selected in each round.

The binary counter argument shows there are at most 2^n distinct stack states. Unfortunately, a specific sequence of box selections can force the algorithm to visit every state. This occurs if we pick box 1 every second step, box 2 every second remaining step, and so on (see Figure 5).

If we measure complexity by unit operations on the stack, this behavior mirrors a binary counter counting from 0 to $2^n - 1$. It is well known that such a counter requires $\Theta(2^n)$ bit-flips to reach the maximum value [2]. Since each bit-flip corresponds to a Push ($0 \to 1$) or a Pop ($1 \to 0$), the worst-case time complexity of Montessori Tower Random Sort is $\Theta(2^n)$.

So how is it possible that Szymon consistently orders $n = 10$ boxes within minutes? Assuming it takes a three-year-old 5 seconds to move one box, the worst-case scenario would require $2 \cdot 1024 \cdot 5 = 10240$ seconds–nearly three hours!

The efficiency Szymon experiences lies in the *expected-time analysis*. If a box is selected uniformly at random from the collection M at each step, then after an expected n selections, the largest box (n) will be picked. Once box n is pushed onto the stack, the stack is cleared of all smaller elements, and box n becomes the permanent bottom of the stack. The problem then reduces to sorting the remaining $n - 1$ boxes. Inductively, the total expected number of PUSH operations is: $n + (n-1) + \ldots + 1 = \frac{1}{2}n^2 + \frac{1}{2}n$. The expected number of POP operations is also $\mathcal{O}(n^2)$. Indeed, for $n = 10$, with 5 seconds per operation, the expected time is roughly $2 \cdot 55 \cdot 5 = 550$ seconds, or just over 9 minutes. This aligns perfectly with the observation that Szymon finishes "within a few minutes."

Comparison with other slow sorting algorithms. Among the most famous "slow" sorting algorithms are BOGO SORT and BOZO SORT. These algorithms perform random updates until a permutation is sorted; BOGO SORT applies a random permutation, while BOZO SORT applies a random transposition. As shown in [3], both algorithms have an average-case time complexity of $\Theta(n!)$. While optimized versions of these algorithms exist with $\mathcal{O}(n^3 \log n)$ complexity [1], the Montessori approach remains naturally more efficient for small n due to its $\mathcal{O}(n^2)$ expected behavior.

5 Conclusion

This work shows that so-called *intuitive* sorting strategies admit rigorous mathematical analysis. We prove that Agata's MAGAZINE SORT NO MARKING is a physically realizable in-place sorting algorithm with worst-case running time $\mathcal{O}(n^2)$. In contrast, Szymon's MONTESSORI TOWER RANDOM SORT exhibits a pronounced separation between worst-case and expected behavior: while its worst-case complexity is exponential, $\Theta(2^n)$, its expected running time is $\mathcal{O}(n^2)$.

For MAGAZINE SORT WITH MARKING, a key complexity question remains unresolved. We establish an $\mathcal{O}(n^3)$ upper bound and observe quadratic behavior in practice; however, determining whether the algorithm admits $\omega(n^2)$ rounds in the worst case remains open.

References

1 Therese C. Biedl, Timothy M. Chan, Erik D. Demaine, Rudolf Fleischer, Mordecai J. Golin, James A. King, and J. Ian Munro. Fun-sort–or the chaos of unordered binary search. *Discret. Appl. Math.*, 144(3):231–236, 2004. doi:10.1016/J.DAM.2004.01.003.

2 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

3 Hermann Gruber, Markus Holzer, and Oliver Ruepp. Sorting the slow way: An analysis of perversely awful randomized sorting algorithms. In Pierluigi Crescenzi, Giuseppe Prencipe, and Geppino Pucci, editors, *Fun with Algorithms, 4th International Conference, FUN 2007, Castiglioncello, Italy, June 3-5, 2007, Proceedings*, volume 4475 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2007. doi:10.1007/978-3-540-72914-3_17.

4 Ian Mertz. Reusing Space: Techniques and Open Problems. *Bull. Eur. Assoc. Theoret. Comput. Sci.*, 141:57–106, 2022.