PAPER

Efficient Algorithms for Sorting k -Sets in Bins*Atsuki NAGAO^{†,††}, Kazuhisa SETO^{†††}, and Junichi TERUYAMA^{††††,†††††a)}, *Members*

SUMMARY We propose efficient algorithms for Sorting k -Sets in Bins. The Sorting k -Sets in Bins problem can be described as follows. We are given numbered n bins with k balls in each bin. Balls in the i -th bin are numbered $n-i+1$. We can only swap balls between adjacent bins. Our task is to move all of the balls to the same numbered bins. For this problem, we give an efficient greedy algorithm with $\frac{k+1}{4}n^2 + O(k+n)$ swaps and provide a detailed analysis for $k=3$. In addition, we give a more efficient recursive algorithm using $\frac{15}{16}n^2 + O(n)$ swaps for $k=3$.

key words: greedy, mathematical puzzles, recursion, sorting, swap

1. Introduction

The sorting problem is a fundamental problem in theoretical computer science, and various types of sorting problems have been studied [5]–[7]. One of the most basic sorting problems is the swap-sort problem as follows: We are given a list with n integers in non-increasing order, and we can only swap two adjacent elements. Our task is to sort them in non-decreasing order. It is well-known that $\binom{n}{2}$ swaps are necessary and sufficient. In this paper, we consider a natural extension of the swap-sort problem. Peter Winkler [15] introduced “Sorting Pairs in Bins,” and Ito, Teruyama and Yoshida [12] extended it to a more general problem, “Sorting k -Sets in Bins”.

Sorting k -Sets in Bins: We are given n numbered bins each with k numbered balls, such that bin i is adjacent to bins $i-1$ and $i+1$, bin n is not adjacent to bin 1, and the balls in bin i are each numbered $n+1-i$. Any two balls may be swapped between adjacent bins. How many swaps are necessary to get every ball into the bin with the matching number?

Manuscript received February 4, 2015.

Manuscript revised June 15, 2015.

Manuscript publicized July 9, 2015.

[†]The author is with Kyoto University, Kyoto-shi, 606–8501 Japan.

^{††}The author is Research Fellow of Japan Society for the Promotion of Science, Tokyo, 102–0083 Japan.

^{†††}The author is with Seikei University, Musashino-shi, 180–8633 Japan.

^{††††}The author is with National Institute of Informatics, Tokyo, 101–8430 Japan.

^{†††††}The author is with JST, ERATO, Kawarabayashi Large Graph Project, Tokyo, 101–8430 Japan.

*The earlier version of this paper appeared in the Proceeding of Eighth International Workshop on Algorithms and Computation (WALCOM 2014), LNCS Volume 8344, pp.225–236.

a) E-mail: teruyama@nii.ac.jp

DOI: 10.1587/transinf.2015EDP7038

For $k=2$, Winkler [15] showed a lower bound $\lceil \binom{2n}{2}/3 \rceil = \lceil \frac{n(2n-1)}{3} \rceil$ and asked whether this lower bound is optimal or not. We can see that n^2 swaps are sufficient using a bubble sort twice. West [16] proposed an algorithm with $\frac{4}{5}n^2$ swaps. Ito, Teruyama and Yoshida [12] gave an affirmative answer to Winkler’s question by designing a nearly optimal algorithm with $\frac{2}{3}n^2 - O(n)$ swaps, and Püttman [14] independently showed a similar result. Ito, Teruyama and Yoshida also gave (without proof) a lower bound for any k and proposed the question of whether there exists an algorithm satisfying the lower bound when $k \geq 3$. When $k=3$, we can easily obtain an algorithm with at most $\frac{3}{2}n^2$ swaps using a bubble sort three times. Moreover, by combining bubble sort with the algorithm in [12], the number of swaps decreases to $\frac{7}{6}n^2$. However, the lower bound is $\frac{9}{10}n^2$. A large gap between the upper bound and the lower bound remains even when $k=3$. Therefore, a natural problem is to design a more efficient algorithm for $k=3$. In this paper, we propose two efficient algorithms, one greedy and one recursive, for this sorting problem for $k=3$ and show that the proposed greedy algorithm is applicable for any k .

1.1 Our Contribution

We propose two algorithms for Sorting 3-Sets in Bins, one of which can be applied to Sorting k -Sets in Bins. We refer to the algorithms as *Greedy*(n, k) and *Recursive*(n). For the Sorting 3-Sets in Bins problem, the maximum number of swaps for *Greedy*($n, 3$) and *Recursive*(n) are $\frac{24}{25}n^2 (= 0.96n^2)$ and $\frac{15}{16}n^2 + O(n) (= 0.9375n^2 + O(n))$, respectively. These values are close to the $\frac{9}{10}n^2 (= 0.9n^2)$ lower bound as shown in [12]. For the Sorting k -Sets in Bins problem, *Greedy*(n, k) achieves $\frac{k+1}{4}n^2 + O(k+n)$ swaps for $k \geq 4$. Formally, we prove the following three theorems.

Theorem 1: There exists a greedy algorithm *Greedy*(n, k) that solves Sorting k -Sets in Bins with $\frac{k+1}{4}n^2 + O(k+n)$ swaps.

Theorem 2: There exists a greedy algorithm *Greedy*($n, 3$) that solves Sorting 3-Sets in Bins with $\frac{24}{25}n^2 + O(n)$ swaps.

Theorem 3: There exists a recursive algorithm *Recursive*(n) that solves Sorting 3-Sets in Bins with $\frac{15}{16}n^2 + O(n)$ swaps.

1.2 Related Work

Few results for our target problem have been reported. However, many other types of sorting problems have been well studied. For example, partial quicksort algorithms address the following problem: Given a list of integers, sort the first l -th smallest elements in the list. The partial quicksort algorithm was introduced by [11] and a tight upper bound was shown in [2]. Another example is the sorting problem for partially ordered sets, in which some pairs of elements are incomparable. Faigle and Tuñan introduced this problem [8] and very recently [4] made substantial advances. For other sorting problems, see e.g. [1], [3], [5]–[7], [9], [10].

1.3 Paper Organization

The remainder of this paper is organized as follows. In Sect. 2, we give some notations required to describe the proposed algorithms. In Sect. 3, we present the *Greedy*(n, k) algorithm and provide an analysis of its performance. In Sect. 4, we present a more efficient algorithm *Recursive*(n) and analyze its performance. Concluding remarks are provided in Sect. 5.

2. Preliminaries

In this section, we provide notations and definitions used to explain the proposed algorithms. To be distinguishable and comparable, we assign all same-numbered balls an index from 1 to k . Let x_i be the ball labeled x and indexed i . We define the total order of balls as $x_i < y_j$ if $x < y$ or if $x = y$ and $i < j$. We represent the state of balls and bins as shown in Fig. 1. The numbers in the bottom row are the bin labels and the other numbers correspond to balls.

The *initial state* of the n bins is the state where the i -th bin has k balls labeled $n + 1 - i$. (Fig. 1.) The *target state* of the n bins is the state where the i -th bin has k balls labeled i . (Fig. 2.) The *solution for n bins* begins with an initial state and ends with a target state for n bins.

If we *move* a ball x_i to the u -th bin, we swap x_i for the lowest-labeled ball in the bin to the right of x_i until x_i arrives at the u -th bin. We know the following fact directly from the properties of “move”.

n_1	$(n-1)_1$		2_1	1_1
n_2	$(n-1)_2$	$\dots$	2_2	1_2
n_3	$(n-1)_3$		2_3	1_3
1	2	$\dots$	$n-1$	n

Fig. 1 Initial state of n bins for $k = 3$

1_1	2_1		$(n-1)_1$	n_1
1_2	2_2	$\dots$	$(n-1)_2$	n_2
1_3	2_3		$(n-1)_3$	n_3
1	2	$\dots$	$n-1$	n

Fig. 2 Target state of n bin as $k = 3$

Fact 1: Suppose that the ball x_i is in the u -th bin and the ball y_j is in the v -th bin with $v < u$. The ball x_i has the minimum label in the u -th bin if and only if the ball x_i shifts to the $(u-1)$ -th bin by moving the ball y_j to the w -th bin (where $w \geq u$).

The following shows the lower bound of Sorting k -Sets in Bins shown in [12]. We have provided a precise proof of Proposition 1 in an earlier version of this article [13].

Proposition 1 ([12]): To solve Sorting k -Sets in Bins, we require at least $\left\lceil \binom{kn}{2} / (2k-1) \right\rceil$ swaps if n is even, and $\left\lceil \left(\binom{kn}{2} - \binom{k-1}{2} \right) / (2k-1) \right\rceil$ swaps if n is odd.

3. Greedy Algorithm

3.1 Greedy(n, k) Algorithm

The proposed greedy algorithm *Greedy*(n, k), which solves Sorting k -Sets in Bins with n bins for any n and k , is shown in Fig. 3.

Lemma 1: The *Greedy*(n, k) algorithm solves Sorting k -Sets in Bins with n bins.

Proof. Let us consider a ball n_i ($i \in [k]$). Initially, it is evident that the n -th bin contains all balls labeled n_i at the end of the loop on line 1 in the *Greedy*(n, k) algorithm when $x = n$. All balls labeled n_i go to the n -th bin on lines 2-4. If a ball n_j ($j \in [k]$) leaves the n -th bin, according to Fact 1, it must be the minimum number in the n -th bin. This situation occurs only when the n -th bin has k balls labeled n_i and j equals 1. This means that the loop on line 1 with $x = n$ is complete. Thus, the n -th bin must contain all balls n_i and the other balls are in the other bins at the end of the loop on line 1 with $x = n$. Next, consider the loop on line 1 with $x = n-1$. By substituting n with $n-1$ and considering the fact that this algorithm does not swap the balls in the $(n-1)$ -th bin with the n -th bin, we obtain a similar result. By repeating this argument, we conclude that *Greedy*(n, k) solves Sorting k -Sets in Bins with n bins correctly. $\square$

3.2 Number of Swaps

Here, we estimate the number of swaps used by *Greedy*(n, k) to solve Sorting k -Sets in Bins with n bins.

Lemma 2: The number of swaps performed by the *Greedy*(n, k) algorithm is at most $\frac{k+1}{4}n^2 + O(k+n)$.

Greedy(n, k), n, k : integer

```

1: for  $x = n$  to 2
2:   for  $i = k$  to 1
3:     Move the ball  $x_i$  to the  $x$ -th bin
4:   end for
5: end for
    
```

Fig. 3 Greedy algorithm

Proof. To prove Lemma 2, we prove the following lemma.

Lemma 3: For any x and any i , where $1 \leq x \leq n$ and $2 \leq i \leq k$, if a ball x_i shifts left from its initial $(n - x + 1)$ -th bin, then x_i has the minimum label of all balls in any bin from $(n - x + 1)$ to n .

Proof. Let us consider the case where a ball x_i shifts left from its initial $(n - x + 1)$ -th bin. From Fact 1, the ball x_i is the minimum-labeled ball in the $(n - x + 1)$ -th bin. Note that $i \geq 2$; therefore the $(n - x + 1)$ -th bin must have a ball y_j , where $y > x$ and $j \in [k]$. Initially, the ball y_j began in the $(n - y + 1)$ -th bin, which is further left than the $(n - x + 1)$ -th bin. In each iteration of the loop on line 1, no balls shift right except when they go to their target bin. Thus, if the above situation occurs, y must be $(n - x + 1)$ and loop iterations from n to $y + 1$ on line 1 are complete. By the proof of Lemma 1, for all $u > n - x + 1$, the u -th bin contains all balls u_l ($l \in [k]$). Thus, the proof is complete. $\square$

First, we count the total number of swaps required for a ball labeled in $[\lfloor n/2 \rfloor + 1, n]$ to go to its target bin.

From Lemma 3, for $\lfloor n/2 \rfloor + 1 \leq x \leq n$ and $2 \leq i \leq k$, a ball x_i does not shift left until it moves to the x -th bin. This means that these x_i balls are in the $(n - x + 1)$ -th bin. The number of swaps required for the ball x_i to move to the x -th bin is $2x - n - 1$. In addition, the number of swaps required for the ball x_1 to move to the x -th bin is trivially at most $x - 1$.

Thus, the total number of swaps required for all balls x_i ($x \in \{\lfloor n/2 \rfloor + 1, \dots, n\}$, $i \in [k]$) to move to their target bins is at most:

$$\begin{aligned} & \sum_{x=\lfloor n/2 \rfloor + 1}^n \{x - 1 + (k - 1) \cdot (2x - n - 1)\} \\ &= \begin{cases} \frac{2k+1}{8}n^2 - \frac{1}{4}n & n : \text{even} \\ \frac{2k+1}{8}n^2 - \frac{1}{2}n + \frac{6k-5}{8} & n : \text{odd}. \end{cases} \end{aligned} \quad (1)$$

Next, we count the total number of swaps required to move the remaining balls to their target bins. To analyze this, we require some lemmas. From Fact 1 and Lemma 3, we obtain Lemma 4 as follows.

Lemma 4: For x , $1 \leq x \leq \lfloor n/2 \rfloor$, and $i \in \{2, \dots, k\}$, suppose that a ball x_i has shifted left at least once and is currently in the l -th bin, where $l < n - x + 1$. Then, all balls in the u -th bin have greater labels than x_i for all $u > l$.

Lemma 5: For x , $1 \leq x \leq \lfloor n/2 \rfloor$, and $i \in \{2, \dots, k\}$, a ball x_i does not shift left from the x -th bin.

Proof. Assume that a ball x_i shifts left from the x -th bin. From Fact 1, ball x_i must have the minimum label in the x -th bin. Moreover, from Lemma 4, all balls in the u -th bin are greater than x_i for all $u > x$. This means that the number of balls that are greater than x_i is at least $(n - x + 1)k - 1$. However, the number of balls that have greater labels than x_i is $(n - x)k + k - i = (n - x + 1)k - i$. Note that this leads

to a contradiction because $i \geq 2$. $\square$

We now have all the tools needed to count the total swaps required to move the remaining balls to their target bins. From the proofs of Lemmas 1 and 5, for $y \leq \lfloor n/2 \rfloor$, $k - 1$ balls y_i ($i \in \{2, \dots, k\}$) are in the y -th bin after the iteration of the loop on line 1 with $x = y$. Thus we only need to move the ball y_1 to complete the loop iteration on line 1 with $x = y$: i.e., the number of swaps required to move ball y_1 is trivially at most $y - 1$. Therefore, in total, the number of swaps required for the loop on line 1 with $y = \lfloor n/2 \rfloor$ to 2 is at most:

$$\sum_{y=2}^{\lfloor n/2 \rfloor} (y - 1) = \begin{cases} \frac{1}{8}n^2 - \frac{1}{4}n & n : \text{even} \\ \frac{1}{8}n^2 - \frac{1}{8} & n : \text{odd}. \end{cases} \quad (2)$$

From Eqs. (1) and (2), in total, the number of swaps *Greedy*(n, k) requires is at most:

$$\begin{cases} \frac{k+1}{4}n^2 - \frac{1}{2}n & n : \text{even} \\ \frac{k+1}{4}n^2 - \frac{1}{2}n + \frac{3k-3}{4} & n : \text{odd}. \end{cases}$$

Thus, Lemma 2 is proved. $\square$

Proof of Theorem 1. The proof of Theorem 1 follows directly from Lemmas 1 and 2. $\square$

3.3 More Exact Analyses for $k = 3$

In this section, we provide a more exact analysis of *Greedy*($n, 3$). In the proof of Lemma 2, we estimate that the number of swaps required for a ball x_1 to move to the x -th bin is trivially at most $x - 1$ for $x \in [n]$. However, some balls with index 1 never go into the first bin. For instance, in the case of $n = 8$, after moving the balls with labels from eight to five, the ball 4_1 is in the second bin. (Fig. 4.) Therefore, the cost for moving the ball 4_1 is only two instead of three on the previous analysis.

We give the following lemma regarding the leftmost position of some balls with index 1.

Lemma 6: For any integer c , if it holds that $x = n - 4c$ and $x - 1 > c + 1$, then, by applying *Greedy*($n, 3$), the balls x_1 and $(x - 1)_1$ do not shift left from the $(c + 1)$ -th bin.

Proof. From Lemma 3, for $\lfloor n/2 \rfloor + 1 \leq y \leq n - c$ and $i = 2, 3$, a ball y_i is in its initial bin, i.e., the $(n - y + 1)$ -th bin until the ball $(n - c + 1)_1$ has moved to the $(n - c + 1)$ -th bin. For instance, in the case of $n = 8$, four balls $5_2, 5_3, 6_2$, and 6_3 are still in their initial bins until the ball 7_1 moves to the seventh bin. (Fig. 5.)

First, consider the following claim.

1 ₁	1 ₃	2 ₂	3 ₂	5 ₁	6 ₁	7 ₁	8 ₁
1 ₂	3 ₁	2 ₃	4 ₂	5 ₂	6 ₂	7 ₂	8 ₂
2 ₁	4 ₁	3 ₂	4 ₃	5 ₃	6 ₃	7 ₃	8 ₃
1	2	3	4	5	6	7	8

Fig. 4 After moving all balls with labels from eight to five for $n = 8$

Claim 1: For any $y \leq x$, after moving the ball $(n - c + 1)_1$ to the $(n - c + 1)$ -th bin, the ball y_1 is in the $(n - y + 1 - 3c)$ -th bin. Moreover, the ball y_1 is always the minimum ball in its bin until ball the $(n - c + 1)_1$ has moved to the $(n - c + 1)$ -th bin.

Proof. At the initial state, the $(n - y + 1)$ -th bin contains the three balls, y_1, y_2 and y_3 . Then, the ball y_1 is the minimum ball in its bin. First, *Greedy*($n, 3$) moves the ball n_3 from the first bin to the n -th bin. From Fact 1, the ball y_1 is shifted to the $(n - y)$ -th bin by moving n_3 to the n -th bin.

The ball y_1 is shifted left by moving the ball n_3 to the n -th bin: i.e., the ball y_1 is in the $(n - y)$ -bin. Now, the $(n - y)$ -th bin contains three balls, y_1 as well as $(y + 1)_2$ and $(y + 1)_3$. Then, the ball y_1 is the minimum ball in the $(n - y)$ -th bin.

Next, *Greedy*($n, 3$) moves the ball n_2 from the first bin to the n -th bin. Then, the ball y_1 is shifted to the $(n - y - 1)$ -th bin. After this operation, the $(n - y - 1)$ -th bin contains three balls, y_1 as well as $(y + 2)_2$ and $(y + 2)_3$. Then, the ball y_1 is also the minimum ball in the $(n - y - 1)$ -th bin.

Inductively, after moving the ball $(n - c + 1)_1$ to the $(n - c + 1)$ -th bin, the ball y_1 is in the $(n - y + 1 - 3c)$ -th bin because the number of shifts of the ball x_1 is $3c$. Note that $y + 3c \leq x + 3c = n - c$, thus, balls $(y + 3c)_2$ and $(y + 3c)_3$ are in the $(n - y + 1 - 3c)$ -th bin. The $(n - y + 1 - 3c)$ -th bin contains the three balls, y_1 , as well as $(y + 3c)_2$ and $(y + 3c)_3$. Then, the ball y_1 is also the minimum ball in the $(n - y + 1 - 3c)$ -th bin. $\square$

From Claim 1, after moving the ball $(n - c + 1)_1$ to the $(n - c + 1)$ -th bin, a ball $(x - i)_1$ is in the $(c + 1 + i)$ -th bin for $i = 0, 1, 2$. The $(c + 1 + i)$ -th bin contains the three balls, $(x - i)_1, (n - c - i)_2$ and $(n - c - i)_3$.

Next, we move the ball $(n - c)_3$ from the $(c + 1)$ -th bin to the $(n - c)$ -th bin. From Fact 1, the ball x_1 does not shift, because x_1 is in the $(c + 1)$ -th bin. Moreover, the ball $(x - 1)_1$ is shifted from the $(c + 2)$ -th bin to the $(c + 1)$ -th bin. Similarly, the ball x_1 does not shift by moving the ball $(n - c)_i$ to the $(n - c)$ -th bin for $i = 3$ to 2 . After moving the ball $(n - c)_2$ to the $(n - c)$ -th bin, the $(c + 1)$ -th bin contains x_1 as well as $(x - 1)_1$ and $(x - 2)_1$.

Next, we move the ball $(n - c)_1$ from the $z (< c + 1)$ -th bin to the $(n - c)$ -th bin. Balls x_1 and $(x - 1)_1$ do not shift by moving the ball $(n - c)_1$ because, as Fact 1 implies, balls x_1 and $(x - 1)_1$ do not have the minimum label in the $(c + 1)$ -th bin.

The rest of the proof shows that balls x_1 and $(x - 1)_1$ are always in the $(c + 1)$ -th bin, i.e., balls x_1 and $(x - 1)_1$ are not the minimum balls in the $(c + 1)$ -th bin after moving the ball $(n - c)_1$ and before moving the ball x_1 .

Now, all balls z_i ($n - c \leq z \leq n$ and $i = 1, 2, 3$) are

2_1	1_1	1_2	1_3	2_2	2_3	7_1	8_1
5_1	3_1	6_2	5_2	4_2	3_2	7_2	8_2
6_1	4_1	6_3	5_3	4_3	3_3	7_3	8_3
1	2	3	4	5	6	7	8

Fig. 5 After moving ball 7_1 for $n = 8$

in the z -th bin. These balls do not shift until *Greedy*($n, 3$) terminates. By Lemma 3 and Lemma 5, the ball z_i ($c + 2 \leq z \leq n - c - 1$ and $i = 2, 3$) does not go to the $(c + 1)$ -th bin. The ball z_1 ($c + 2 \leq z \leq n - c - 1$) is in the $z' (\leq c)$ -th bin. Since the ball z_i , which is greater than $(x - 1)_1$, does not shift from the $(c + 2)$ -th bin to the $(c + 1)$ -th bin, balls x_1 and $(x - 1)_1$ do not have the minimum label in the $(c + 1)$ -th bin.

Thus, the proof is complete. $\square$

Lemma 7: The number of swaps performed by the *Greedy*($n, 3$) algorithm is at most $\frac{24}{25}n^2 + O(n)$.

Proof. Lemma 6 implies that the number of swaps required to move x_1 and $(x - 1)_1$ is at most $x - c - 1$ and $x - c - 2$ rather than $x - 1$ and $x - 2$, respectively.

Then, we can reduce the bound for x_1 with $(x - 1) - (x - c - 1) = c$. In addition, we can reduce the bound for $(x - 1)_1$ with $(x - 2) - (x - c - 2) = c$. From the conditions $x = n - 4c$ and $x - 1 \geq c + 1$, $c \leq \lfloor \frac{n-2}{5} \rfloor$ holds. Thus, the total reduction of the upper bound for *Greedy*($n, 3$) is at least:

$$\sum_{c=1}^{\lfloor \frac{n-2}{5} \rfloor} 2c = \left\lfloor \frac{n-2}{5} \right\rfloor \cdot \left(\left\lfloor \frac{n-2}{5} \right\rfloor - 1 \right) \geq \frac{1}{25}n^2 - n + 6.$$

The last inequality holds from $\lfloor \frac{n-2}{5} \rfloor > \frac{n}{5} - 2$. In a simple analysis, the upper bound for *Greedy*($n, 3$) is at most $n^2 + O(n)$. Then, the total number of swaps of *Greedy*($n, 3$) is at most:

$$n^2 + O(n) - \left(\frac{1}{25}n^2 - n + 6 \right) = \frac{24}{25}n^2 + O(n).$$

Therefore, the proof is complete. $\square$

Proof of Theorem 2. Theorem 2 follows directly from Lemma 1 and Lemma 7. $\square$

4. Recursive Algorithm

4.1 Outline of Proposed Recursive Algorithm

Here, we present a recursive algorithm that is more efficient for the case $k = 3$. Before describing the details of the proposed recursive algorithm, we outline it using a case where n is a multiple of 6 as an example.

First, we move the ball n_3 from the first bin to the n -th bin using $n - 1$ swaps. Next, we move the ball n_2 from the first bin to the n -th bin using $n - 1$ swaps, and the state becomes as shown in Fig. 6. Then, we move the ball $(n - 1)_3$ from the second bin to the $(n - 1)$ -th bin using $n - 3$ swaps, and we move the ball $(n - 1)_2$ from the second bin to the

$(n - 2)_1$	$(n - 3)_1$	$(n - 4)_1$	$\dots$	1_1	1_2	1_3
$(n - 1)_1$	$(n - 1)_2$	$(n - 2)_2$	$\dots$	3_2	2_2	n_2
n_1	$(n - 1)_3$	$(n - 2)_3$	$\dots$	3_3	2_3	n_3
1	2	3	$\dots$	$n - 2$	$n - 1$	n

Fig. 6 After moving balls n_3 and n_2 to the n -th bin

$(n-1)$ -th bin using $n-3$ swaps. Now, the state is as shown in Fig. 7. For $x = n$ to $n/2 + 1$, we similarly move the ball labeled x_3 to the x -th bin. We then move the ball labeled x_2 to the x -th bin. Now, the resulting state is as shown in Fig. 8.

Here, for the first $n/3$ bins (i.e., from the first bin to the $(n/3)$ -th bin), we label the ball x_1 as $\lceil x/3 \rceil$, where $x \in [n]$. Using this state as the initial state of the $n/3$ bins problem, we can proceed recursively. After the recursive call, returning the labels of the balls in the first $n/3$ bins to their original labels yields a state whereby the first bin contains balls 1_1 , 2_1 and 3_1 , the second bin contains balls 4_1 , 5_1 and 6_1 , and so on (Fig. 9).

Finally, for $x = n$ to 2 , we move ball x_1 to the x -th bin as follows. First, when we move the ball n_1 from the $(n/3)$ -th bin to the n -th bin, balls $1_2, 4_2, 7_2, \dots, (n/2-2)_2$ and $n/2_3, (n/2-1)_3, \dots, 1_3$ are shifted left (Fig. 10). Next, when moving the ball $(n-1)_1$ from the $(n/3)$ -th bin to the $(n-1)$ -th bin, balls $2_2, 5_2, 8_2, \dots, (n/2-1)_2$ and $(n/2-1)_3, \dots, 1_3$ are shifted left (Fig. 11). Similarly, we move balls $(n-2)_1, (n-3)_1, \dots, 2_1$ to their respective target bins in order and sorting is complete.

The same strategy can be applied for n that are not multiples of 6. The details are shown in the next subsection.

$(n-2)_1$	$(n-5)_1$	$(n-6)_1$	$\dots$	2_2	2_3	1_3
$(n-1)_1$	$(n-4)_1$	$(n-2)_2$	$\dots$	3_2	$(n-1)_2$	n_2
n_1	$(n-3)_1$	$(n-2)_3$	$\dots$	3_3	$(n-1)_3$	n_3
1	2	3	$\dots$	$n-2$	$n-1$	n

Fig. 7 After moving balls $(n-1)_3$ and $(n-1)_2$ to the $(n-1)$ -th bin

4.2 Recursive(n) algorithm

The *Recursive(n)* algorithm, which is presented in Fig. 12, solves for n bins with $k = 3$.

Here, we prove the correctness of the *Recursive(n)* algorithm.

Lemma 8: The *Recursive(n)* algorithm solves for n bins.

Proof. We divide the algorithm into three steps as follows.

Recursive(n), n : integer

```

1: for  $x = n$  to  $\lceil n/2 \rceil + 1$ 
2:   Move the ball  $x_3$  to the  $x$ -th bin
3:   Move the ball  $x_2$  to the  $x$ -th bin
4: end for
5: for  $x = 1$  to  $n$ 
6:   Relabel the ball  $x_1$  (in the 1st bin to the  $\lceil n/3 \rceil$ -th bin) to  $\lceil x/3 \rceil$ 
7: end for
8: If the  $\lceil n/3 \rceil$ -th bin has any ball  $y_2$  or  $y_3$  ( $y \in [n]$ )
9: then Relabel this ball with  $\lceil n/3 \rceil$ 
10: Solve for the first  $\lceil n/3 \rceil$  bins by applying Recursive( $\lceil n/3 \rceil$ )
    (By lines 5–9, all balls in the first  $\lceil n/3 \rceil$  bins are numbered
    at most  $\lceil n/3 \rceil$ . Therefore the first  $\lceil n/3 \rceil$  bins are reduced to
    an instance of Sorting 3-Sets in Bins with  $\lceil n/3 \rceil$  bins.)
11: relabel all balls in the first  $\lceil n/3 \rceil$  bins with their original numbers
12: for  $x = n$  to  $2$ 
13:   Move the ball  $x_1$  to the  $x$ -th bin
14: end for

```

Fig. 12 Recursive algorithm

$(n-2)_1$	$\dots$	1_1	1_2	4_2	$\dots$	$(n/2-2)_2$	$n/2_3$	$\dots$	2_3	1_3
$(n-1)_1$	$\dots$	2_1	2_2	5_2	$\dots$	$(n/2-1)_2$	$(n/2+1)_2$	$\dots$	$(n-1)_2$	n_2
n_1	$\dots$	3_1	3_2	6_2	$\dots$	$n/2_2$	$(n/2+1)_3$	$\dots$	$(n-1)_3$	n_3
1	$\dots$	$n/3$	$n/3+1$	$n/3+2$	$\dots$	$n/2$	$n/2+1$	$\dots$	$n-1$	n

Fig. 8 After moving balls x_3 and x_2 to the x -th bin for $x = n$ to $n/2 + 1$

1_1	4_1	$\dots$	$(n-2)_1$	1_2	$\dots$	$(n/2-2)_2$	$n/2_3$	$\dots$	2_3	1_3
2_1	5_1	$\dots$	$(n-1)_1$	2_2	$\dots$	$(n/2-1)_2$	$(n/2+1)_2$	$\dots$	$(n-1)_2$	n_2
3_1	6_1	$\dots$	n_1	3_2	$\dots$	$n/2_2$	$(n/2+1)_3$	$\dots$	$(n-1)_3$	n_3
1	2	$\dots$	$n/3$	$n/3+1$	$\dots$	$n/2$	$n/2+1$	$\dots$	$n-1$	n

Fig. 9 The state after sorting for recursive structure

1_1	$\dots$	1_2	2_2	$\dots$	$(n/2-1)_2$	$(n/2-1)_3$	$\dots$	1_3	n_1
2_1	$\dots$	$(n-2)_1$	3_2	$\dots$	$n/2_2$	$(n/2+1)_2$	$\dots$	$(n-1)_2$	n_2
3_1	$\dots$	$(n-1)_1$	4_2	$\dots$	$n/2_3$	$(n/2+1)_3$	$\dots$	$(n-1)_3$	n_3
1	$\dots$	$n/3$	$n/3+1$	$\dots$	$n/2$	$n/2+1$	$\dots$	$n-1$	n

Fig. 10 After moving the ball n_1 to the n -th bin

1_1	$\dots$	1_2	3_2	$\dots$	$(n/2-1)_3$	$(n/2-2)_3$	$\dots$	$(n-1)_1$	n_1
2_1	$\dots$	2_2	4_2	$\dots$	$n/2_2$	$(n/2+1)_2$	$\dots$	$(n-1)_2$	n_2
3_1	$\dots$	$(n-2)_1$	5_2	$\dots$	$n/2_3$	$(n/2+1)_3$	$\dots$	$(n-1)_3$	n_3
1	$\dots$	$n/3$	$n/3+1$	$\dots$	$n/2$	$n/2+1$	$\dots$	$n-1$	n

Fig. 11 After moving the ball $(n-1)_1$ to the $(n-1)$ -th bin

...	$(n - \lceil i/2 \rceil + 1)_2$	...	x_1	...
...	$(n - \lceil i/2 \rceil + 1)_3$	...	$(x + i - 1)_2$	...
...	$\lceil i/2 \rceil$	...	$(x + i - 1)_3$	...
...		...	$n - x + 2 - i$	...

Fig. 13 Before the i -th movement when i is odd

Step 1 is shown in lines 1–4, Step 2 is shown in lines 5–11 and Step 3 is shown in lines 12–14 (Fig. 12). We show which bins have which balls after each step.

[Step 1] We consider three cases, where each case considers the index of the balls (e.g., x_1 , x_2 and x_3).

- A.** Let x_1 be an arbitrary ball with index 1 and $x \in [n]$. Let m be the number of times the ball x_1 moves left. Note that the ball x_1 is initially in the $(n - x + 1)$ -th bin. Here, we consider what happens when we move balls $(n - y + 1)_3$ or $(n - y + 1)_2$ from the y -th bin to the $(n - y + 1)$ -th bin for each y . If the ball x_1 is located in the u -th bin, where $u > y$, then x_1 shifts left once during this movement. In the i -th movement, when i is odd, we move the ball $(n - \lceil i/2 \rceil + 1)_3$ in the $\lceil i/2 \rceil$ -th bin. When i is even, we move the ball $(n - \lceil i/2 \rceil + 1)_2$ in the $\lceil i/2 \rceil$ -th bin. The ball x_1 is in $(n - x + 2 - i)$ -th bin if $\lceil i/2 \rceil < n - x + 2 - i$ holds. (Fig. 13) Then, m is the largest integer that satisfies $\lceil m/2 \rceil < n - x + 2 - m$. This is expressed as follows:

$$m = \max\{0 \leq i \leq n - 1 \mid i + \lceil i/2 \rceil < n - x + 2\}.$$

After iterating $2 \cdot \lfloor n/2 \rfloor$ times, the ball x_1 goes to the $(n - x + 1 - m)$ -th bin.

We set r and q such that $n - x + 2 = 3r + q$, $q \in \{0, 1, 2\}$. Then, m is $2r - 1$ if $q = 0$ and $2r$ if $q = 1, 2$. After Step 1, the ball x_1 is in the bin numbered as follows:

$$n - x + 1 - m = 3r + q - 1 - m = \begin{cases} r & (q = 0, 1) \\ r + 1 & (q = 2) \end{cases}.$$

As $x = n - 3r - q + 2$, the r -th bin contains the three balls $(n - 3r + 1)_1$, $(n - 3r + 2)_1$, and $(n - 3r + 3)_1$ after Step 1, where $x \in [n]$, $1 \leq r \leq \lceil n/3 \rceil$.

- B.** Let x_2 be an arbitrary ball with index 2 and $x \in \{1, \dots, \lfloor n/2 \rfloor\}$. Using analysis that is similar to the proof of Lemma 3, we obtain the following fact.

Fact 2: The ball x_2 does not shift before the ball $(n - x + 1)_2$ moves to the $(n - x + 1)$ -th bin in Step 1.

Here, we consider what happens to x_2 when we move balls $(n - y + 1)_2$ or $(n - y + 1)_3$ from the y -th bin to the $(n - y + 1)$ -th bin, where $n - y + 1 \leq n - x + 1$. If the ball x_2 is in the u -th bin, where $u > y$, x_2 shifts left during this movement. By a similar argument to case A, the ball x_2 goes to the $(n + x - m)$ -th bin, where m is the maximum integer that satisfies $\lceil m/2 \rceil < (n - x + 1) - (m - 1 - 2x + 1)$:

$$m = \max\{0 \leq i \leq n - 1 \mid i + \lceil i/2 \rceil < n + x + 1\}.$$

We set $n + x + 1 = 3r + q$, $q \in \{0, 1, 2\}$. By calculation, m is $2r - 1$ if $q = 0$ and $2r$ if $q = 1, 2$. Thus, after Step 1, the ball x_2 is in the following bin:

$$n + x - m = 3r + q - 1 - m = \begin{cases} r & (q = 0, 1) \\ r + 1 & (q = 2) \end{cases}.$$

As $x = 3r + q - n - 1$, the r -th bin contains the three balls $(3r - n - 2)_2$, $(3r - n - 1)_2$, and $(3r - n)_2$ after Step 1, where $\lceil (n + 1)/3 \rceil \leq r \leq \lfloor n/2 \rfloor$.

C. Any ball x_3 ($x = 1, \dots, \lfloor n/2 \rfloor$) does not shift.

[Step 2] Applying the recursive algorithm, balls x_1 ($x = 1, \dots, n$) are sorted in the first $\lfloor n/3 \rfloor$ bins. Each ball x_1 is in the $\lceil x/3 \rceil$ -th bin.

[Step 3] Here we observe the state after moving the ball n_1 to the n -th bin. Note that the ball n_1 is in the $\lfloor n/3 \rfloor$ -th bin after Step 2.

Balls of the form x_1 ($x = 1, \dots, n - 1$) do not shift, because their bin numbers are less than or equal to $\lfloor n/3 \rfloor$.

Balls of the form x_2 ($x = 1, \dots, \lfloor n/2 \rfloor$) are each in a bin between $\lceil (n + 1)/3 \rceil$ and $\lfloor n/2 \rfloor$ after Step 2. From the analysis of Step 1 for x_2 , the r -th bin has three balls $(3r - n - 2)_2$, $(3r - n - 1)_2$, and $(3r - n)_2$. As the ball $(3r - n - 2)_2$ shifts left once while the ball n_1 moves to the right, the r -th bin will have the three balls $(3r - n - 1)_2$, $(3r - n)_2$, and $(3r - n + 1)_2$ after completing this movement.

Before moving ball n_1 , each ball of the form x_3 ($x = 1, \dots, \lfloor n/2 \rfloor$) is in the $(n - x)$ -th bin, which is not the target bin. These balls each shift left once as the ball n_1 moves to the n -th bin. In other words, every ball x_3 ($x = 1, \dots, \lfloor n/2 \rfloor$) shifts to the $(n - x - 1)$ -th bin.

From the above arguments, the state of the first $n - 1$ bins after moving the ball n_1 to the n -th bin is the same as the state after Step 2 when we apply *Recursive*($n - 1$) to *Sorting 3-Sets in Bins* with $n - 1$ bins. After the second to last execution of the Step 3 loop (when $x = 3$), the state of the first two bins is the same as the state after Step 2 in the solution of *Sorting 3-Sets in Bins* with two bins. In other words, the state is as follows:

1_1	1_3
1_2	2_2
2_1	2_3
1	2

We can complete the sorting by moving 2_1 to the second bin. $\square$

4.3 Number of Swaps

Here, we count the number of swaps for the *Recursive*(n) algorithm.

Lemma 9: The number of swaps performed by *Recursive*(n) is less than $\frac{15}{16}n^2 + 12n$.

Proof. Here, we use induction. Let $S(n)$ be the number of

swaps performed by *Recursive*(n). When $n = 2$, $S(2) = 3 < \frac{15}{16} \cdot 2^2 + 24$.

Suppose that $S(l) < \frac{15}{16}l^2 + 12l$ holds for any $l < n$. We count the number of swaps performed by *Recursive*(n), where $n \geq 3$.

In Step 1, we require $2i - n - 1$ swaps to move each ball i_3 and i_2 to the i -th bin; thus, the total number of swaps in Step 1 is at most

$$\sum_{i=n}^{\lceil n/2 \rceil + 1} (2i - n - 1) \times 2 = 2 \cdot \lceil n/2 \rceil \cdot (n - \lceil n/2 \rceil) \leq \frac{1}{2}n^2.$$

Step 2 requires $S(\lceil n/3 \rceil) < \frac{15}{16}(\lceil n/3 \rceil)^2 + 12\lceil n/3 \rceil$ swaps.

Note that we must also count the number of swaps in Step 3. Here, we consider the state of each bin. The r -th bin ($r \in \{1, \dots, \lceil n/3 \rceil\}$) has three balls each labeled $(3r - 3 + j)_1$ ($j \in \{1, 2, 3\}$.) The total number of swaps required for these three balls to move to the target bin is at most

$$\sum_{j=1}^3 (3r - 3 + j - r) = 6r - 3.$$

Therefore, Step 3 requires at most

$$\sum_{r=1}^{\lceil n/3 \rceil} 6r - 3 = 3(\lceil n/3 \rceil)^2$$

swaps. Thus,

$$\begin{aligned} S(n) &< \frac{1}{2}n^2 + \frac{15}{16}(\lceil n/3 \rceil)^2 + 12\lceil n/3 \rceil + 3(\lceil n/3 \rceil)^2 \\ &< \frac{15}{16}n^2 + 12n, \end{aligned}$$

is required. $\square$

Proof of Theorem 3. Theorem 3 follows directly from Lemmas 8 and 9. $\square$

5. Concluding Remarks

In this paper, we have investigated Sorting k -set in Bins.

Fig. 14 Number of swaps for *Greedy*($n, 3$)

First, we presented the *Greedy*(n, k) algorithm, which requires at most $\frac{k+1}{4}n^2 + O(k+n)$ swaps for all k . Moreover, we have shown that *Greedy*($n, 3$) requires at most $\frac{24}{25}n^2 + O(n)$ swaps by a more exact analysis. Second, we introduced the *Recursive*(n) algorithm, which solves for only $k = 3$, but performs better than *Greedy*($n, 3$). The number of swaps required for *Recursive*(n) is at most $\frac{15}{16}n^2 + O(n)$.

In the future, we plan to analyze the number of swaps for *Greedy*(n, k) tightly. We have simulated *Greedy*($n, 3$) in C++ and have plotted the results in Fig. 14. The results indicate that the analysis of *Greedy*($n, 3$) is not tight and the simulation of *Greedy*($n, 3$) performs nearly $\frac{15}{16}n^2$ swaps. In addition, we intend to improve the proposed recursive algorithm by making it applicable to any $k \geq 4$.

Acknowledgments

This work was supported in part by ELC project (Grant-in-Aid for Scientific Research on Innovative Areas MEXT Japan, KAKENHI No. 24106003). This research was also supported by JST, ERATO, Kawarabayashi Large Graph Project, and Japan Society for the Promotion of Science. The authors would like to thank Lyle Waldman for his help in editing the earlier version of this paper.

References

- [1] M. Bóna and R. Flynn, "Sorting a Permutation with Block Moves," arXiv:0806.2787v1.
- [2] Conrado Martínez and Uwe Röser, Partial quicksort and quick-partitionsort, DMTCS Proceedings 01, pp.505–512, 2010.
- [3] D.W. Cranston, I.H. Sudborough, and D.B. West, "Short Proofs for Cut-and-Paste Sorting of Permutations," *Discrete Mathematics*, vol.307, no.22, pp.2866–2870, 2007.
- [4] C. Daskalakis, R.M. Karp, E. Mossel, S.J. Riesenfeld, and E. Verbin, "Sorting and selection in posets," *SIAM Journal on Computing*, vol.40, no.3, pp.597–622, 2011.
- [5] D.J. Kleitman, E. Kramer, J.H. Conway, S. Bell, and H. Dweighter, "Elementary Problems," *American Mathematical Monthly*, vol.82, no.10, pp.1009–1010, 1975.
- [6] S. Elizalde and P. Winkler, "Sorting by Placement and Shift," *Proc. ACM/SIAM Symp. on Discrete Algorithms (SODA)*, pp.68–75, 2009.
- [7] H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, and J. Wästlund, "Sorting a bridge hand," *Discrete Math*, vol.241, no.1-3 pp.289–300, 2001.
- [8] U. Faigle and G. Turán, "Sorting and Recognition Problems for Ordered Sets," *SIAM Journal on Computing*, vol.17, no.1, pp.100–113, 1988.
- [9] W.H. Gates and C.H. Papadimitriou, "Bounds for sorting by prefix reversal," *Discrete Math*, vol.27, no.1 pp.47–57, 1979.
- [10] M.H. Heydari and I.H. Sudborough, "On the diameter of pancake network," *J. Algorithms*, vol.25, no.1, pp.67–94, 1997.
- [11] C.A.R. Hoare, PARTITION (Algorithm 63); QUICKSORT (Algorithm 64); FIND (Algorithm 65), *Communication of the Association for Computing Machinery*, 4, pp.321–322, 1961.
- [12] H. Ito, J. Teruyama, and Y. Yoshida, "An almost optimal algorithm for winkler's sorting pairs in bins," *Progress in Informatics*, no.9, pp.3–7, 2012.
- [13] A. Nagao, K. Seto, and J. Teruyama, "Efficient Algorithms for Sorting k -Sets in Bins," In *Proceedings of the 8th International Workshop on Algorithms and Computation (WALCOM 2014)*, LNCS

8344, pp.225–236, vol.8344, pp.225–236, 2014 (Chennai, India).

- [14] A. Püttmann, “KRAWATTENPROBLEM,” <http://www.springer.com/cda/content/document/cda/downloaddocument/SAV`Krawattenraetsel`Loesung`Puettmann>
- [15] P. Winkler, *Mathematical Puzzles: A Connoisseur’s Collection*, pp.143, 149–151, A K Peters, 2004.
- [16] D.B. West, <http://www.math.uiuc.edu/~west/regs/sortpair.html>, 2008.

Atsuki Nagao is a student of Graduate School of Informatics, Kyoto University. He received B.Eng. and M.Info. degrees from Kyoto University in 2010 and 2012 respectively.

Kazuhisa Seto received B.Eng. and M.Info. and Dr. of Informatics degrees from Kyoto University in 2005, 2007, and 2010 respectively. Since 2013, he has been an assistant professor in Graduate School of Science and Technology at Seikei University.

Junichi Teruyama is a project researcher in Global Research Center for Big Data Mathematics at National Institute of Informatics. He is also a researcher in JST, ERATO, Kawarabayashi Large Graph Project. He received B.Eng., M.Info. and Dr. of Informatics degrees from Kyoto University in 2008, 2010, and 2013 respectively.