Special issue: Theoretical computer science and discrete mathematics

Research Paper

An almost optimal algorithm for Winkler's sorting pairs in bins

Hiro ITO[1], Junichi TERUYAMA[2], and Yuichi YOSHIDA[3]
[1,2] *School of Informatics, Kyoto University, Japan*
[3] *School of Informatics, Kyoto University, and Preferred Infrastructure, Inc., Japan*

ABSTRACT

We investigate the following sorting problem: We are given n bins with two balls in each bin. Balls in the ith bin are numbered $n+1-i$. We can swap two balls from adjacent bins. How many number of swaps are needed in order to sort balls, i.e., move every ball to the bin with the same number. For this problem the best known solution requires almost $\frac{4}{5}n^2$ swaps. In this paper, we show an algorithm which solves this problem using less than $\frac{2n^2}{3}$ swaps. Since it is known that the lower bound of the number of swaps is $\left\lceil \binom{2n}{2}/3 \right\rceil = \left\lceil \frac{2n^2}{3} - \frac{n}{3} \right\rceil$, our result is almost tight. Furthermore, we show that for $n = 2^m + 1$ ($m \geq 0$) the algorithm is optimal.

KEYWORDS

Bubble sort, mathematical puzzle, recursion, sorting, swap

1 Introduction

Sorting is a fundamental operation in computer science, and many types of sorting problems have been developed. A well-studied model of sorting, e.g., bubble sort, merge sort, quick sort, and so on, is based on a comparison sort. These sort algorithms are based on comparing two numbers and swapping them. Some versions of sorting problem have been studied [3]–[5].

In this paper, we investigate the following sorting problem, which we call SORTING k-SETS IN BINS. This problem was posed by Peter Winkler [12].

> Standing in a row are n bins, the ith bin containing k balls numbered $n+1-i$. At any time, we can swap two balls from adjacent bins. How many swaps are necessary to get every ball into the bin carrying its number?

If $k = 1$, this problem is a well-known standard swap-sort problem, and it is easy to see that $\binom{n}{2}$ swaps are necessary and sufficient. In this paper, we consider the case of $k = 2$.

We see an example for the case of $n = 3$, we are given three bins with two balls respectively as Fig. 1, our task is replacing balls to target state as Fig. 2 by swapping two balls from adjacent bins. From the result of $k = 1$, you may think $2 \cdot \binom{3}{2} = 6$ swaps are necessary. However, five swaps are sufficient (see in Fig. 3)! In fact we can easily check that less than $2 \cdot \binom{n}{2}$ swaps are sufficient for small n by using computer search.

Winkler showed that $\left\lceil \binom{2n}{2}/3 \right\rceil = \left\lceil \frac{n(2n-1)}{3} \right\rceil$ is a lower bound [12]. This bound looks nice since it is tight for $n \leq 5$. However for $n = 6$, we need $23 > 22 = \left\lceil \binom{2\cdot 6}{2}/3 \right\rceil$ swaps, and hence the bound is not tight generally. Finding a tight bound and finding an algorithm giving the bound is an attractive open problem, and we don't have any non trivial upper bound so far. (However, in the website [13], it is described that an upper bound of $\frac{4}{5}n^2$ is obtained by recursively using the fact the sorting can be done in 15 moves for $n = 5$).

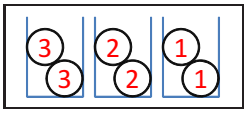


Fig. 1 Initial state.

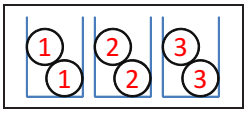


Fig. 2 Target state.

Received October 30, 2011; Revised December 20, 2011; Accepted December 22, 2011.
[1] *itohiro@kuis.kyoto-u.ac.jp*, [2] *teruyama@kuis.kyoto-u.ac.jp*, [3] *yyoshida@kuis.kyoto-u.ac.jp*
DOI: 10.2201/NiiPi.2012.9.2

©2012 National Institute of Informatics

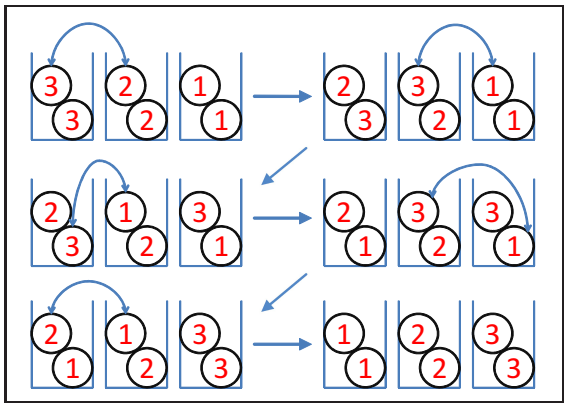


Fig. 3 5 swaps to sort 3 bins.

Our contribution. Let $T(n)$ be the minimum number of swaps for SORTING PAIRS IN BINS with n bins. We show that $T(n)$ is less than $\frac{2}{3}n^2$. Since it is known that the lower bound of the number of swaps is $\left\lceil \binom{2n}{2}/3 \right\rceil = \left\lceil \frac{2n^2}{3} - \frac{n}{3} \right\rceil$ [12], our result is almost tight. In particular, we show that $T(n) = \binom{2n}{2}/3$ for $n = 2^m + 1$ ($m = 0, 1, 2, \ldots$), which matches the lower bound [12]. Main approach of our algorithm is recursivity to solve SORTING PAIRS IN BINS. We can find a recursive structure of this problem to reach our upper bound. Finding this structure gives the effect for $n = 2^m + 1$ bins, and one can solve in optimal swaps.

That is, we show the following result.

Theorem 1. *There is an algorithm that solves* SORTING PAIRS IN BINS *less than* $\frac{2}{3}n^2$ *swaps for* $n \geq 2$ *bins. In particular, if* $n = 2^m + 1$ ($m = 0, 1, 2, \ldots$)*, it sorts in* $\binom{2n}{2}/3$ *swaps and this is optimal.*

Püttman independently reported a similar result in [11]. However, we have informally reported the present work in [8], [9] earlier. Also, we cannot confirm the detail of her paper since the manuscript is written in German.

Related work. Consider the case of $k = 1$, that is, each bin has one ball. Any permutation can be sorted in at most $\binom{n}{2}$ exchanges. A well-known method is bubble sort [10]. There is no better bound, because when the balls start in reverse order each ball must be exchanged with every other to complete the sorting.

Some versions of sorting problem have been studied besides a comparison sort. For example, a permutation is modeled as pancakes [3], [6], [7] (sorting operation is flipping the prefix of permutation). Others of sorting operation are cut-and-paste or block transposition [1], [2], [4] (sorting strings of consecutive entries of a permutation). And one may sort by placement and shift as hand-sorting files [5].

n	$n-1$	$\cdots$	2	1
n	$n-1$	$\cdots$	2	1

Fig. 4 Simple representation of the state of balls and bins.

$n-1$	$n-2$	$n-3$	$\cdots$	2	1	1
n	$n-1$	$n-2$	$\cdots$	3	2	n

Fig. 5 The state after moving a ball n to the nth bin from the initial state.

$n-1$	$n-3$	$n-4$	$\cdots$	1	2	1
n	$n-2$	$n-2$	$\cdots$	3	$n-1$	n

Fig. 6 The state after moving the rightmost ball $n-1$ to the $(n-1)$th bin from the state as Fig. 5.

2 Algorithm

2.1 Outline of our algorithm

Before explaining our algorithm, we give some notations. An *initial state* for n bins is the state that any ith bin has two balls labeled $n+1-i$ as Fig. 1. A *target state* for n bins is the state that any ith bin has two balls labeled its number i as Fig. 2. We define that to *solve for n bins* is to sort balls from the initial state to the target state for n bins. If we say *move* the ball u to the vth bin, one continues to swap u and the lower number ball in u's right bin until the ball u goes into the vth bin. In this paper, we represent the state of balls and bins simply as Fig. 4 instead of Fig. 1.

First, we explain the outline of our algorithm with the case that n (the number of bins) is even as follows. First, we move one ball n from the 1st bin to the nth bin by using $n-1$ swaps, and the state becomes as Fig. 5. Next, we move the rightmost ball $n-1$ from the 2nd bin to the $(n-1)$th bin by $n-3$ swaps, and the state becomes as Fig. 6. Similarly we move the rightmost ball $n-i$ from the $(i+1)$th bin to the $(n-i)$th bin for $i = 0$ to $n/2-1$. Now, the resulting state is as Fig. 7.

Here, looking the $n/2$ bins of the left half, we relabel balls i by $\left\lceil \frac{i}{2} \right\rceil$ for $i \in \{1, \ldots, n\}$. The state of $n' = \frac{n}{2}$ bins in the left half is regarded as the initial state of n' bins. Thus we can use this recursive structure, and we solve for n' bins, and the resulting state of n' bins in the left half becomes the target state of n' bins. Returning

$n-1$	$n-3$	$\cdots$	3	1	$\frac{n}{2}$	$\cdots$	3	2	1
n	$n-2$	$\cdots$	4	2	$\frac{n}{2}+1$	$\cdots$	$n-2$	$n-1$	n

Fig. 7 The left half bins is a recursive structure: the state of after Step 1 of algorithm $PB(n)$ when n is even.

2	4	$\cdots$	$n-2$	n	$\frac{n}{2}$	$\cdots$	3	2	1
1	3	$\cdots$	$n-3$	$n-1$	$\frac{n}{2}+1$	$\cdots$	$n-2$	$n-1$	n

Fig. 8 The state after sorting for a recursive structure: the state of after Step 2 of $PB(n)$ when n is even.

2	4	$\cdots$	$n-2$	$n-1$	$\frac{n}{2}-1$	$\cdots$	2	1	n
1	3	$\cdots$	$n-3$	$\frac{n}{2}$	$\frac{n}{2}+1$	$\cdots$	$n-2$	$n-1$	n

Fig. 9 The state after moving the left most ball n to the nth bin from the state as Fig. 8.

2	4	$\cdots$	$n-2$	$\frac{n}{2}-1$	$\frac{n}{2}-2$	$\cdots$	1	$n-1$	n
1	3	$\cdots$	$n-3$	$\frac{n}{2}$	$\frac{n}{2}+1$	$\cdots$	$n-2$	$n-1$	n

Fig. 10 The state after moving the left most ball $n-1$ to the $(n-1)$th bin from the state as Fig. 9.

balls in n' bins in the left half to the original number, the state of n' bins in the left half is that the 1st bin has balls 1 and 2, the 2nd bin has balls 3 and 4, and so on. Thus the resulting state is as Fig. 8.

As the last step, for $i = 0$ to $n-2$, we move the leftmost ball $n-i$ to the $(n-i)$th bins as follows. First, when we move the leftmost ball n from the $\frac{n}{2}$th bin to the nth bin, the rightmost ball $n/2$ is shifted to the $(n/2)$th bin, and each rightmost ball $n/2-1,\ldots,1$ is shifted to left bin. The resulting state is as in Fig. 9. Next, moving a leftmost ball $n-1$ from the $\frac{n}{2}$th bin to the $(n-1)$th bin, each rightmost ball $n/2-1,\ldots,1$ is shifted to left bin. The state is shown in Fig. 10. Similarly we move balls $n-2, n-3,\ldots,2$ to the target state bins respectively in this order, the resulting state becomes the target state of n bins.

For odd n, this strategy can be applied by introducing a fine adjustment. The details are shown in the next subsection.

$n-1$	$n-3$	$\cdots$	2	1	$\frac{n-1}{2}$	$\cdots$	3	2	1
n	$n-2$	$\cdots$	3	$\frac{n+1}{2}$	$\frac{n+3}{2}$	$\cdots$	$n-2$	$n-1$	n

Fig. 11 The state of after Step 1 of $PB(n)$ when n is odd.

2	4	$\cdots$	$n-1$	n	$\frac{n-1}{2}$	$\cdots$	3	2	1
1	3	$\cdots$	$n-2$	$\frac{n+1}{2}$	$\frac{n+3}{2}$	$\cdots$	$n-2$	$n-1$	n

Fig. 12 The state of after Step 2 of $PB(n)$ when n is odd.

2.2 Algorithm

We present an algorithm $PB(n)$ that solves for n bins as follows.

Algorithm $PB(n)$.

- when n is even
  1. For $i = 0$ to $n/2-1$, move the rightmost ball with labeled $n-i$ to the $(n-i)$th bin. After this operation, the resulting state is as Fig. 7.
  2. If $n = 2$ do nothing. Otherwise looking the $n/2$ bins in the left half, relabel balls i by $\lceil \frac{i}{2} \rceil$ for $i \in \{1,\ldots,n\}$. Its state is the same as the initial state of $n' = n/2$ bins problem. Solve for n' bins in the left half by $PB(n')$. For balls in n' bins in the left half, we replace these labels back to the original. After that, the state is shown in Fig. 8.
  3. For $i = 0$ to $n-2$, move the leftmost ball $n-i$ to the $(n-i)$th bin.
- when n is odd
  1. For $i = 0$ to $(n-3)/2$, move the rightmost ball $n-i$ to the bin $n-i$. After this operation, the resulting state is as Fig. 11.
  2. Looking the $(n+1)/2$ bins in the left half, relabel i by $\lceil \frac{i}{2} \rceil$ for $i \in \{1,\ldots,n\}$ except the rightmost ball $(n+1)/2$. Now the state of $n'' = (n+1)/2$ bins in the left is as Fig. 5 for n'' bins. In fact, this state is on the way of Step 1 in $PB(n'')$ for n'' bins. Therefore, one can sort balls from this state to the target state of n'' bins by simulating $PB(n'')$ from the state. For balls in n'' bins in the left half, we replace these labels back to the original. After that, the resulting state is shown in Fig. 12.
  3. For $i = 0$ to $n-2$, move the leftmost ball $n-i$ to the $(n-i)$th bin.

Lemma 1. *The algorithm $PB(n)$ solves for n bins.*

Proof. Clear from the induction. □

3 Number of swaps

To prove Theorem 1, it remains to show algorithm $PB(n)$ uses less than $\frac{2}{3}n^2$ swaps.

Lemma 2. *The number of swaps performed by $PB(n)$ is less than $\frac{2n^2}{3}$.*

Proof. Let $S(n)$ be the number of swaps performed by $PB(n)$. We use induction. When $n = 2$, $S(2) = 2 < \frac{2 \cdot 2^2}{3}$ clearly holds.

Suppose that $S(l) < \frac{2l^2}{3}$ holds for any $l < n$. There are two cases to consider.

- When n is even:
  We count the number of swaps performed by $PB(n)$. In Step 1, $\frac{n^2}{4}$ swaps are needed. Step 2 needs $S\left(\frac{n}{2}\right) < \frac{2}{3}\left(\frac{n}{2}\right)^2$ swaps. Finally, Step 3 needs $\frac{n^2}{4}$ swaps. Thus, in total, less than $\frac{n^2}{4} + \frac{2}{3}\left(\frac{n}{2}\right)^2 + \frac{n^2}{4} = \frac{2n^2}{3}$ swaps are required.
- When n is odd:
  We count the number of swaps performed by $PB(n)$. In Step 1, $\frac{n^2-1}{4}$ swaps are needed. Step 2 needs $S\left(\frac{n+1}{2}\right) - \frac{n-1}{2} < \frac{2}{3}\left(\frac{n+1}{2}\right)^2 - \frac{n-1}{2} = \frac{n^2-n+4}{6}$ swaps. And Step 3 needs $\frac{n^2-1}{4}$ swaps. Thus, in total, less than $\frac{n^2-1}{4} + \frac{n^2-n+4}{6} + \frac{n^2-1}{4} = \frac{2n^2}{3} - \frac{n-1}{6} < \frac{2n^2}{3}$ swaps are required.

Therefore we complete the proof. □

Lemma 3. *When $n = 2^m + 1$, the number of swaps performed by $PB(n)$ is exactly $\binom{2n}{2}/3$.*

Proof. We use induction on m. When $m = 0$, we can clearly sort in $\binom{2(2^0+1)}{2}/3 = 2$ swaps.

Let $n = 2^m + 1$ and suppose that $PB(n)$ performs $\binom{2n}{2}/3$ swaps. Let $n' = 2^{m+1} + 1$. From the argument on PB in the proof of Lemma 2, the number of swaps performed by $PB(n')$ is

$$\begin{aligned}
&\frac{n'^2-1}{4} + S\left(\frac{n'+1}{2}\right) - \frac{n'-1}{2} + \frac{n'^2-1}{4} \\
&= \frac{n'^2-1}{4} + \binom{2n}{2}\Big/3 - \frac{n'-1}{2} + \frac{n'^2-1}{4} \\
&= \binom{2n'}{2}\Big/3.
\end{aligned}$$

□

Proof of Theorem 1. Directly follows from Lemmas 1, 2, and 3. □

4 Conclusions

We showed that there is an algorithm for solving the problem of SORTING PAIRS IN BINS less than $\frac{2}{3}n^2$ swaps and especially, it achieves the optimal for $n = 2^m + 1$. Note that this problem considers only reverse (double) sequence for the input. However it is not difficult to show that this case gives the worst case among the problem considering all permutations for input.

We conjecture that for any n but 6 it is possible to solve by $\left\lceil \frac{n(2n-1)}{3} \right\rceil$ swaps, which matches the lower bound. That is because our computer search found a sequence of swaps in steps equal to lower bound for $n \leq 20$ except 6. Therefore, one of our future works is to find an algorithm performing the lower bound for every n but 6.

In addition we have an open question that if each bin contains more than two balls, i.e., SORTING k-SETS IN BINS, how many times one needs to swap balls. We can calculate the lower bound of the number of swaps by using Winkler's "point system" as well as the case that each bin has two balls. If each bin has k balls, the lower bound of the number of swaps is $\left\lceil \binom{kn}{2}/(2k-1) \right\rceil$ if n is even, and $\left\lceil \left(\binom{kn}{2} - \binom{k-1}{2}\right)/(2k-1) \right\rceil$ if n is odd.

References

[1] M. Bóna and R. Flynn, "Sorting a permutation with block moves," arXiv:0806.2787v1.

[2] D. Cranston, I. H. Sudborough, and D. B. West, "Short proofs for cut-and-paste sorting of permutations," *Discrete Math.*, vol.307, pp.2866–2870, 2007.

[3] H. Dweighter, "Elementary Problems," *American Mathematical Monthly*, vol.82, p.1010, 1975.

[4] H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, and J. Wástlund, "Sorting a bridge hand," *Discrete Math.*, vol.241, pp.289–300, 2001.

[5] S. Elizalde and P. Winkler, "Sorting by Placement and Shift," *Proc. ACM/SIAM Symp. on Discrete Algorithms (SODA)*, pp.68–75, 2009.

[6] W. H. Gates and C. H. Papadimitriou, "Bounds for sorting by prefix reversal," *Discrete Math.*, vol.27, pp.47–57, 1979.

[7] M. H. Heydari and I. H. Sudborough, "On the diameter of pancake network," *J. Algorithms*, vol.25, pp.67–94, 1997.

[8] H. Ito, J. Teruyama, and Y. Yoshida, "An Almost Optimal Algorithm for Winkler's Sorting Pairs in Bins", *IEICE Computation*, *IEICE Technical Report*, vol.109, no.391, COMP2009-45, pp.45–49, 2010.

[9] H. Ito, J. Teruyama, and Y. Yoshida, "An Almost Optimal Algorithm for Winkler's Sorting Pairs in Bins", *Proc. of the 3rd Asian Association for Algorithms and Computation (AAAC 2010)*, p.11, 2010.

[10] D. E. Knuth, *Sorting and Searching, volume 3 of The Art of Computer Programming*, Addison-Wesley, 1973, Second edition, 1998.

[11] A. Püttmann, "KRAWATTENPROBLEM", http://www.springer.com/cda/content/document/cda_downloaddocument/SAV_Krawattenraetsel_Loesung_Puettmann

[12] P. Winkler, *Mathematical puzzles: A Connoisseur's collection*. A K Peters, vol.143, pp.149–151, 2004.

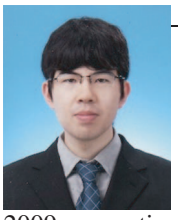
[13] D. B. West: http://www.math.uiuc.edu/~west/regs/sortpair.html, 2008.

Hiro ITO

Hiro ITO received the B.E., M.E., and Dr. of Engineering degrees in the Department of Applied Mathematics and Physics from the Faculty of Engineering, Kyoto University in 1985, 1987, and 1995, respectively. From 1987 to 1996 and from 1996 to 2001 he was a member of NTT Laboratories and Toyohashi University of Technology, respectively. Since 2001, he has been an associate professor in the Department of Communications and Computer Engineering, Graduate School of Informatics at Kyoto University. He has been engaged in research on discrete algorithms mainly on graphs and networks, discrete mathematics, and recreational mathematics. Dr. Ito is a member of IEICE, the Operations Research Society of Japan, the Information Processing Society of Japan, and the European Association for Theoretical Computer Science.

Junichi TERUYAMA

Junichi TERUYAMA is a student of Graduate School of Informatics, Kyoto University. He received B.Eng. and M.Info. degrees from Kyoto University in 2008 and 2010, respectively. His main research interests include quantum computing.



Yuichi YOSHIDA

Yuichi YOSHIDA is a student of Graduate School of Informatics, Kyoto University. He cofounded Preferred Infrastructure Inc. in 2006. He received B.Eng. and M.Info. degrees from Kyoto University in 2007 and 2009, respectively. His main research interests include approximation algorithms, randomized algorithms and property testing.