Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Discrete Mathematics 36 (2010) 231–238

Electronic Notes in DISCRETE MATHEMATICS

www.elsevier.com/locate/endm

# Pancake Flipping with Two Spatulas

Mahfuza Sharmin [a,b,1], Rukhsana Yeasmin [b,2], Masud Hasan [b,3], Atif Rahman [c,4] and M. Sohel Rahman [b,d,5]

[a] *Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka-1208, Bangladesh*

[b] *Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh*

[c] *Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA*

[d] *Algorithm Design Group, Department of Computer Science, King's College London, Strand, London WC2R 2LS, England*

**Abstract**

In this paper, we give approximation algorithms for several variations of the *pancake flipping problem*, which is also well known as the problem of *sorting by prefix reversals*. We consider the variations in the sorting process by adding prefix transpositions, prefix transreversals etc. along with the prefix reversals.

*Keywords:* Approximation algorithms, genome sorting, pancake flipping, sorting permutations

[1] Email: `mhfz.sharmin@gmail.com`
[2] Email: `smrity_23@yahoo.com`
[3] Email: `masudhasan@cse.buet.ac.bd`
[4] Email: `atif@eecs.berkeley.edu`
[5] Email: `msrahman@cse.buet.ac.bd`

doi:10.1016/j.endm.2010.05.030

## 1 Introduction

Given a permutation $\pi$, a *reversal* flips a substring of $\pi$, a *transposition* cuts a substring of $\pi$ and pastes it in a different location, and a *transreversal* is a transposition with a flip done before the substring is pasted. In a *prefix* version of each operation the corresponding substring is always a prefix of $\pi$.

The *pancake flipping problem* [3,7,13,16,17] deals with finding the minimum number of prefix reversals required to sort a given permutation. This problem was first introduced in 1975 by Dweighter [7], who described the motivation of a chef to rearrange a stack of pancakes.

Aside from being an interesting combinatorial problem, this problem and its variations have applications in interconnection networks and computational biology. The number of flips required to sort the stack of $n$ pancakes is the *diameter* of the $n$-dimensional *pancake network* [16,17]. A well studied variation of pancake flipping problem is the *burnt pancake flipping problem* [3,16,17].

A broader class of similar sorting problems, called the *genome rearrangement problems*, is extensively studied in computational molecular biology. In order to explain the existence of essentially the same set of genes having differences only in their order in different species, several rearrangement operations have been suggested, including reversals [1,4,19], transpositions [2,8,14,10], transreversals [15], prefix transposition [6], prefix reversal [11], etc.

This type of sorting problems are mostly NP-complete or of unknown complexities. Caprara [5] proved that sorting by reversals is NP-hrad, whereas Heydari and Sudborough [18] have claimed that sorting by prefix reversals is NP-complete. Hence, researchers have focused mainly on different approximation algorithms for these problems and their variantions. A number of authors have also considered these by using more than one operations (reversals, transpositions etc.) [9,12,15,20,21,22,23], mostly for signed permutations.

In this paper we study some variations of pancake flipping problem from sorting permutations view point. We give a 3-approximation algorithm for sorting by prefix reversals and prefix transpositions and a 2-approximation algorithm for sorting by prefix reversals and prefix transreversals.

We also introduce the concept of *forward march*, the idea of which comes naturally from a greedy approach where someone may try to sort from starting to end, and give a 3-approximation algorithm for this problem.

The above problems resembles to variants of the original pancake flipping problem where the chef has two spatulas in his two free hands. He can either lift some pancakes from the top of the stack and flip them (a prefix reversal) or he can lift a top portion of the stack with one hand, lift another portion from the top with the other hand, and place the top portion under the second portion (a prefix transposition), possibly with a flip (a prefix transreversal). Also, time to time, when a top portion of the stack is sorted he can remove it from the stack (a forward march).

It is worth mentioning that the worst case ratios of our algorithms can only be realized when an optimal algorithm applies no prefix reversals at all, which is very unlikely in practice. Keeping this observation in mind, we derive mathematically the equations for *adaptive* approximation ratio in terms of the number of prefix reversals applied by an optimal algorithm.

We organize the rest of the paper as follows. Section 2 gives the preliminaries. Section 3 presents the approximation algorithms. In Section 4, we derive equations for approximation ratio in terms of number of prefix reversals applied by an optimal algorithm. Finally, Section 5 concludes the paper. Notably, for space constraints we skipped the proofs of the lemmas and theorems, which will be provided in the journal version.

## 2 Preliminaries

Let $\pi = [\pi_0, \pi_1, \ldots, \pi_n, \pi_{n+1}]$ be a permutation of $n + 2$ distinct elements, where $\pi_0 = 0$, $\pi_{n+1} = n + 1$, and for each $1 \leq i \leq n$, $1 \leq \pi_i \leq n$; the middle $n$ elements of $\pi$ are to be sorted. A *prefix reversal* $\beta = \beta(1, j)$ for some $3 \leq j \leq n+1$ transforms $\pi$ into $\pi \cdot \beta = [\pi_0, \pi_{j-1}, \ldots, \pi_1, \pi_j, \ldots, \pi_{n+1}]$. A *prefix transposition* $\tau = \tau(1, j, k)$ for some $2 \leq j \leq n$ and $3 \leq k \leq n + 1$ with $k \notin [1, j]$, transforms $\pi$ into $\pi \cdot \tau = [\pi_0, \pi_j, \ldots, \pi_{k-1}, \pi_1, \ldots, \pi_{j-1}, \pi_k, \ldots, \pi_{n+1}]$.

An *identity permutation* $\iota_n$ is a permutation such that $\pi_i = i$ for all $1 \leq i \leq n$. Given two permutations, the *problem of sorting* one permutation to another is equivalent to the problem of sorting a given one to $\iota_n$ [1,2]. The *prefix reversal and prefix transposition distance* between $\pi$ and $\iota$ is $d(\pi) = t$ such that $\pi \cdot o_1 \cdot \ldots \cdot o_{d(\pi)} = \iota$ and $t$ is minimum, where for all $1 \leq i \leq t$ either $o_i = \beta$ or $o_i = \tau$. Given a permutation $\pi$, the problem of *sorting by prefix reversals and prefix transpositions* is to find $d(\pi)$.

A *breakpoint* for this problem is a position $i$ in $\pi$ such that $2 \leq i \leq n$ and $|\pi_i - \pi_{i-1}| \neq 1$. In addition, position 1 is always considered as a breakpoint. Position $n+1$ is considered a breakpoint when $\pi_n \neq n$. Let, $b(\pi)$ is the number of breakpoints in $\pi$. Therefore, $b(\pi) \geq 1$, and $b(\pi) = 1$ only when $\pi = \iota_n$.

The *breakpoint graph* $G_\pi$ of $\pi$ is an undirected multi graph whose vertices are $\pi_i$, for $0 \leq i \leq n + 1$, and edges are of two types: *grey* and *black*; for $1 \leq i \leq n+1$, the vertices $\pi_i$ and $\pi_{i-1}$ are joined by a black edge if and only if there is a breakpoint between them, and for $0 \leq j < i \leq n + 1$ and $j \neq i - 1$,

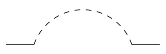
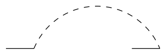
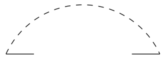
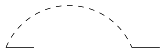
| | Edge Type | Scenario |
|---|---|---|
| 1 |  | $\pi_0 - \pi_1 \ldots \pi_{i-1} - \pi_i \ldots \pi_j - \pi_{j+1} \xrightarrow{\tau_1} \pi_0 - \pi_i \ldots \pi_j \pi_1 \ldots \pi_{i-1} - \pi_{j+1}$ |
| 2 |  | $\pi_0 - \pi_1 \ldots \pi_{j-1} - \pi_j \xrightarrow{\beta_2} \pi_0 - \pi_{j-1} \ldots \pi_1 \pi_j$ |
| 3 |  | $\pi_0 - \pi_1 \ldots \pi_i - \pi_{i+1} \ldots \pi_{j-1} - \pi_j \xrightarrow{\tau_3} \pi_0 - \pi_{i+1} \ldots \pi_{j-1} - \pi_1 \ldots \pi_i \pi_j$ |
| 4 |  | $\pi_0 - \pi_1 \ldots \pi_{i-1} - \pi_i \ldots \pi_{j-1} - \pi_j \xrightarrow{\beta_4} \pi_0 - \pi_{j-1} \ldots \pi_i - \pi_{i-1} \ldots \pi_1 - \pi_j$ |

Fig. 1. Edge types and Scenarios of SortByRT3.

there is a grey edge between $\pi_i$ and $\pi_j$ if and only if $|\pi_i - \pi_j| = 1$.

## 3 Approximation Algorithms

### *3.1 A 3-approximation algorithm for prefix reversals and prefix transpositions*

For a permutation $\pi$ and an operation $o$, denote $\triangle(\pi, o) = b(\pi) - b(\pi \cdot o)$ as the number of breakpoints that are removed due to operation $o$. Some obvious but important observations about breakpoints are that $\triangle(\pi, \beta) \leq 1$ and $\triangle(\pi, \tau) \leq 2$. That means, an optimal algorithm for this problem can not remove more than two breakpoints by a single operation, which implies a lower bound of $d(\pi) \geq \lfloor \frac{b(\pi)-1}{2} \rfloor$.

Our algorithm (let us call it *SortByRT3*) works on considering different orientations of grey and black edges. Note that if a permutation is not sorted, then there must be at least two grey edges in $G(\pi)$ and each grey edge will be incident to two black edges. A grey edge with its two adjacent black edges must be one of the four types as shown in Fig. 1.

**Lemma 3.1** *Let $(\pi_1, \pi_j)$ be a Type 1 grey edge. Then there exists at least one black edge $(\pi_{i-1}, \pi_i)$ for some $2 \leq i \leq j$. We call such a black edge a* trapped black edge*.*

In our algorithm we scan the permutation from left to right to find the first black edge incident to a grey edge. We apply the four scenarios for each edge type, when possible, in the order as shown in Fig. 1.

**Lemma 3.2** *Given $\pi$ and $G(\pi)$, if any of the following two conditions is satisfied, then a prefix reversal or a prefix transposition can be applied to $\pi$ such that it removes at least one breakpoint: (1) $G(\pi)$ contains a grey edge*

*$(\pi_1, \pi_j)$ of Type 1 or 2 with $\pi_1 \neq 1$, and (2) $G(\pi)$ contains a grey edge $(\pi_i, \pi_j)$ of Type 3 with $\pi_1 = 1$.*

**Lemma 3.3** *Given $\pi$ and $G(\pi)$, if Scenario 1, 2 or 3 is not applicable, then a prefix reversal can be applied that does not remove any breakpoint but is followed by two subsequent operations that removes at least two breakpoints.*

**Theorem 3.4** *SortByRT3 is a 3-approximation algorithm.*

### *3.2 A 2-approximation algorithm*

Now we improve the ratio considering *prefix transreversal.* A *prefix transreversal* $\beta\tau = \beta\tau(1, j, k)$, for some $2 \leq j \leq n$, $3 \leq k \leq n+1$ with $k \notin [1, j]$, transforms $\pi$ into $\pi \cdot \beta\tau = [\pi_0, \pi_j, \ldots, \pi_{k-1}, \pi_{j-1}, \ldots, \pi_1, \pi_k, \ldots, \pi_{n+1}]$. Another important observation about breakpoints regarding prefix transreversals is: $\triangle(\pi, \beta\tau) \leq 2$, which keeps the lower bound for sorting by prefix reversals and prefix transreversals same as: $d(\pi) \geq \lfloor \frac{b(\pi)-1}{2} \rfloor$. Now, the next lemma is the key to our 2-approximation.

**Lemma 3.5** *Let $\pi$ be a permutation with $\pi_1 = 1$ and let its associated breakpoint graph be $G(\pi)$. If $G(\pi)$ contains a grey edge of Type 4, then a prefix transreversal can be applied that removes at least one breakpoint.*

**Theorem 3.6** *An algorithm (SortByRT2) that produces prefix reversals, prefix transpositions, and/or prefix transreversals according to Lemma 3.5 is an approximation algorithm with factor 2 for sorting by prefix reversals and prefix transreversals.*

### *3.3 A 3-approximation algorithm with forward march*

At the very beginning or after applying a prefix reversal/transposition to $\pi$, a prefix $\pi_0, \ldots, \pi_i$, for $0 \leq i \leq n+1$, may be already sorted. In that case we update $\pi$ as the unsorted suffix of $\pi$, i.e., as $\pi = \pi_i, \ldots, \pi_{n+1}$, and the size of $\pi$ is reduced by $i$, i.e., $n = n - i$. The next prefix reversal or prefix transposition is applied on updated $\pi$. This concept of moving forward along with the sorting is called *forward march.*

For our algorithm with forward march, we redefine breakpoint and breakpoint graph. We no more consider the position 1 as a default breakpint, Clearly, $\pi$ is sorted if and only if it has no breakpoint. Note that at any time, $\pi_0$ is the last element in the sorted part and there always exists a black edge between $\pi_0$ and $\pi_1$. We call this black edge the *starting black edge.*

Due to breakpoint redefinition, some of our previous observations are modified. In particular, now we have $\triangle(\pi, \beta) \leq 2$ and $\triangle(\pi, \tau) \leq 3$. Consequently, a new lower bound is: $d(\pi) \geq \frac{b(\pi)}{3}$.

Our algorithm (SortByRTwFM3) works on considering different orientation of grey and black edges (see Fig. 1). The unsorted $\pi t$ has at least two grey edges and at least one black edge in addition to the starting black edge $(\pi_0, \pi_1)$. We apply a prefix transposition or a prefix reversal in order according to the five scenarios shown in Fig. 2, and if possible, perform a forward march. It is immediate that after each reversal or transposition the number of breakpoints is reduced by at least one. So our algorithm needs at most $b(\pi)$ operations. With the lower bound of $d(\pi) \geq \frac{b(\pi)}{3}$, the following theorem holds.

**Theorem 3.7** *SortByRTwFM3 is a 3-approximation algorithm.*

Now, next Lemma proves that Scenarios 4 and 5 can complete sorting and others are to improve the practical performance without affecting the ratio.

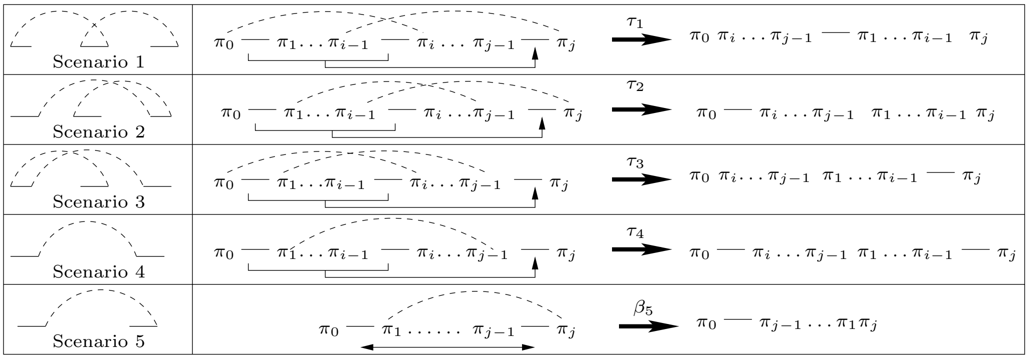

Scenario 1
$\pi_0$ — $\pi_1 \ldots \pi_{i-1}$ — $\pi_i \ldots \pi_{j-1}$ — $\pi_j$
$\tau_1$
$\pi_0$ $\pi_i \ldots \pi_{j-1}$ — $\pi_1 \ldots \pi_{i-1}$ $\pi_j$
Scenario 2
$\pi_0$ — $\pi_1 \ldots \pi_{i-1}$ — $\pi_i \ldots \pi_{j-1}$ — $\pi_j$
$\tau_2$
$\pi_0$ — $\pi_i \ldots \pi_{j-1}$ $\pi_1 \ldots \pi_{i-1}$ $\pi_j$
Scenario 3
$\pi_0$ — $\pi_1 \ldots \pi_{i-1}$ — $\pi_i \ldots \pi_{j-1}$ — $\pi_j$
$\tau_3$
$\pi_0$ $\pi_i \ldots \pi_{j-1}$ $\pi_1 \ldots \pi_{i-1}$ — $\pi_j$
Scenario 4
$\pi_0$ — $\pi_1 \ldots \pi_{i-1}$ — $\pi_i \ldots \pi_{j-1}$ — $\pi_j$
$\tau_4$
$\pi_0$ — $\pi_i \ldots \pi_{j-1}$ $\pi_1 \ldots \pi_{i-1}$ — $\pi_j$
Scenario 5
$\pi_0$ — $\pi_1 \ldots\ldots \pi_{j-1}$ — $\pi_j$
$\beta_5$
$\pi_0$ — $\pi_{j-1} \ldots \pi_1 \pi_j$

Fig. 2. Scenarios of SortByRTwFM3

**Lemma 3.8** *Scenario 4 and Scenario 5 are sufficient to sort the permutation.*

## 4 Adaptive approximation ratios

Motivated by observation that, an optimal algorithm would apply both prefix reversal and prefix transposition we derive approximation ratio $\rho_r$ that is *adaptive* to the number of prefix reversals $r$ applied by an optimal algorithm.

**Theorem 4.1** *When $r$ prefix reversals are applied by respective optimal algorithms, adaptive approximation ratios of our algorithms are: (1) $\rho_r \leq 3 - \frac{3r}{b(\pi)+r-1}$ for SortByRT3, (2) $\rho_r \leq 2 - \frac{2r}{b(\pi)+r-1}$ for SortByRT2, and (3) $\rho_r \leq 3 - \frac{3r}{b(\pi)+r}$ for SortByRTwFM3.*

## 5 Conclusion

In this paper we have studied some variations of pancake flipping problem from sorting unsigned permutations view point, problems that arise in genome rearrangement. Besides the algorithm with performance ratio 3 and 2, we have introduced a new concept called forward march which contributes another 3-approximation algorithm. By analyzing the problems in a more practical way better approximation ratios have been presented. In future it would be interesting to improve the approximation ratio and to decide their complexities.

## References

[1] Bafna, V., and Pevzner, P., *Genome rearrangements and sorting by reversals*, Proc. FOCS'93 (1993), 148-157. Also in *SIAM J. Comp.*, **25** (1996) 272-289.

[2] Bafna, V., and Pevzner, P., *Sorting Permutations by Transpositions*, Proc. SODA'95 (1995), 614-623. Also in *SIAM J. Discr. Math.*, **11(2)** (1998) 224-240.

[3] Bass, D.W., and Sudborough, I.H., *Pancake problems with restricted prefix reversals and some corresponding Cayley networks*, J. Paral. Distr. Comp. **63(3)** (2003), 327-336.

[4] Berman, P., Hannenhalli, S., and Karpinski, M., *1.375-approximation algorithm for sorting by reversals*, Proc. ESA'02 in:LNCS 2461, Springer (2002), 200-210.

[5] Caprara, A., *Sorting by Reversals is difficult*, Proc. RECOMB'97 (1997), 75-83.

[6] Dias, Z., and Meidanis, J., *Sorting by Prefix Transpositions*, Proc. SPIRE'02, in:LNCS 2476, Springer (2002), 463-468.

[7] Dweighter, H., *Problem E2569*, Amer. Math. Monthly **82** (1975), 1010.

[8] Elias, I., and Hartman, T., *A 1.375-Approximation Algorithm for Sorting by Transpositions*, Proc. WABI'05 in:LNCS 3962, Springer (2005), 204-214.

[9] Eriksen, N., *(1+$\epsilon$)-approximation of Sorting by Reversals and Transpositions*, Theor. Comp. Sci., **289(1)** (2002), 517-529.

[10] Firoz, J.S., Hasan, M., Khan, A.Z., Rahman, M. Sohel, *The 1.375 Approximation Algorithm for Sorting by Transpositions Can Run in $O(n \log n)$ Time*, Proc.WALCOM 2010 in:LNCS 5942, Springer (2010), 161-166.

[11] GATES, W.H., and PAPADIMITRIOU, C.H., *Bounds for sorting by prefix reversal*, Discrete Mathematics, **27** (1979), 47-57.

[12] Gu, Q.P., Peng, S., and Sudborough, H., *A 2-approximation algorithm for genome rearrangements by reversals and transpositions*, Theor. Comp. Sci., **210(2)** (1999), 327-339.

[13] Hannenhalli, S., and Pevzner, P., *Transforming cabbage into turnip*, Proc. STOC'95 **(1995)**, 178-189.

[14] Hartman, T., *A simpler 1.5-approximation algorithm for sorting by transpositions*, Proc. CPM'03 in:LNCS 2676, Springer (2003), 156-169.

[15] Hartman, T., and Sharan, R., *A 1.5-approximation algorithm for sorting by transpositions and transreversals*, Proc.WABI'04 in:LNCS 3240, Springer (2004), 50-61.

[16] Heydari, M.H., and Sudborough, I.H., *On Sorting by Prefix Reversals and the Diameter of Pancake Networks*, Parallel Architectures and Their Efficient Use, in:LNCS 678, Springer (1993), 218-227.

[17] Heydari, M.H., and Sudborough, I.H., *On the diameter of the pancake network*, J. Algorithms, **25(1)** (1997), 67-94.

[18] Heydari, M.H., and Sudborough, I.H., *Sorting by prefix reversals is NP-complete*, To be submitted (as mentioned in [23]).

[19] Kececioglu, J., and Sankoff, D., *Exact and approximation algorithms for the inversion distance between two permutations*, Proc. CPM'93, in:LNCS 648 Springer (1993), 87-105. Also in *Algorithmica*, 13 (1995)180-210.

[20] Lin, G.H., and Xue, G., *Signed genome rearrangements by reversals and transpositions: Models and approximations*, Proc. COCOON'99 in:LNCS 1672, Springer (1999), 71-80.

[21] Lou, X., and Zhu, D., *A 2.25-Approximation Algorithm for Cut-and-Paste Sorting of Unsigned Circular Permutations*, Proc. COCOON'08, in:LNCS 2476, Springer (2008), 331-341.

[22] Rahman, A., Shatabda, S., and Hasan, M., *An Appoximation Algorithm for Sorting by Reversals and Transpositions*, J. Discr. Algorithms **6(3)** (2008), 449-457.

[23] Walter, M.E., Dias, Z., and Meidanis, J., *Reversal and transposition distance of linear chromosomes*, Proc. SPIRE'98 in:IEEE CS, **3962** (1998), 96-102.