The Pancake Problem:
Pre x Reversals of Certain Permutations

Alyssa Armstrong

May 8, 2009

Contents

1 Abstract

The Pancake Problem concerns the minimum number of moves needed to order a random stack of di erently-sized pancakes. Mathematically, this

Figure 1: Flipping a Stack of Pancakes

We will use the one line notation.

Definition 2.3. The **identity permutation** $\iota \in S_n$ maps each element of the set $\{1, 2, \ldots, n\}$ to itself. Thus, in our one line notation, for $\iota \in S_n$, $\iota = (1\ 2\ 3 \ldots n)$.

Also, each time the chef flips a stack of pancakes, a portion of the permutation is reversed. We can define this flip as a prefix reversal:

Definition 2.4. Given $\pi \in S_n$, a **prefix reversal at** π_i of $\pi = (\pi_1\ \pi_2 \ldots \pi_i \ldots \pi_n)$ is $\pi' \in S_n$ such that $\pi' = (\pi_i \ldots \pi_2\ \pi_1\ \pi_{i+1} \ldots \pi_n)$.

Example 2.5. Let $\pi \in S_n$, such that $\pi = (4\ 7\ 2\ 1\ 5\ 3\ 6)$. The prefix reversal of π at 5 is $\pi' = (5\ 1\ 2\ 7\ 4\ 3\ 6)$.

Thus, the Pancake Problem translates to conducting prefix reversals on a permutation until the identity permutation is achieved.

3 Initial Algorithm

After experimenting with a few small permutations, one can create a trivial algorithm to find the minimum number of reversals needed to obtain the identity permutation.

Lemma 3.1. *The lower bound for the number of reversals needed to transform a permutation, $\pi \in S_n$, to the identity is at most $2n$ reversals.*

Proof. We show this using the following algorithm:

1. Given $\pi \in S_n$, reverse at the largest number that is not in its sorted position. (Note: a number is in its sorted position when $\pi_i = i$.)

2. Reverse so that number is in its sorted position.

3. Repeat steps 1 and 2 until the identity permutation is achieved.

Since it takes at most two reversals to sort each element of π to its sorted position, it will take at most $2n$ reversals to transform π to ι. □

Example 3.2. Given the permutation, $\pi = (3\ 1\ 5\ 4\ 2)$. Following the trivial algorithm,

1. Doing step 1 of the algorithm, we reverse the permutation at 5 to obtain (5 1 3 4 2).

2. Doing step 2 of the algorithm, we reverse the permutation at 2 to obtain (2 4 3 1 5).

3. Doing step 1 of the algorithm, we reverse the permutation at 4 to obtain (4 2 3 1 5).

4. Doing step 2 of the algorithm, we reverse the permutation at 1 to obtain (1 3 2 4 5).

5. Doing step 1 of the algorithm, we reverse the permutation at 3 to obtain (3 1 2 4 5).

6. We do not need to do step 1 of the algorithm, so we reverse the permutation at 2 to obtain (2 1 3 4 5).

7. We do not need to do step 1 of the algorithm, so we reverse the permutation at 1 to obtain (1 2 3 4 5).

Therefore, it takes 7 reversals to transform to . This is less than the maximum of 10 because we did not need to reverse two times when sorting 1 and 2. It is common for this algorithm to result in fewer than $2n$ reversals in practice.

4 Gates' Algorithm

As an undergraduate at Harvard University in 1979, Bill Gates was presented the Pancake Problem in his Combinatorial Mathematics class as an example of a problem that was simple to propose, but di cult to solve. In just a few days, Gates returned to his professor, claiming that he had created a general algorithm in order to rearrange a permutation $2 S_n$. Gates and his advisor, Christos Papadimitriou, decreased the lower bound of reversals from $2n$ to $\frac{5n+5}{3}$ $1{:}667n$, by classifying a permutation based on its block structure and creating an algorithm that will transform any $2 S_n$ to . What follows are a few de nitions about his block structure.

De nition 4.1. Given the permutation, $2 S_n$.

If $j \ {}_i \quad {}_j j \quad 1$ (*mod n*), then i is **consecutive** to j.

If $j \ {}_i \quad {}_{i+1} j = 1$, then the pair $(i; i+1)$ is an **adjacency** in .

A **block** is a maximal length sublist, $x = \ {}_i \ {}_{i+1} \colon\colon\colon \ {}_{j-1} \ {}_j = y$, such that there is an adjacency between ${}_a$ and ${}_{a+1}$ for all $i \quad a \quad j$. We +

our classi cation of is B C_{-} A $_{-}$ D.

Gates and Papadimitriou thus de ne an algorithm which classi es a permutation into one of nine cases based on the structure of the initial element and its consecutive elements (shown below). Once the case is identi ed for a permutation, the detailed reversals are performed creating a newly arranged permutation. This process is repeated until the identity permutation is achieved.

Example 4.3. Suppose we are given $2\ S_7$ where $= (2\ 3\ 4\ 7\ 6\ 1\ 5)$.

By Gate's Algorithm,

1. The permutation begins with the block $(2; 3; 4)$ where and 2 is consecutive to

Gates' Algorithm - Reversal Sequences		
Case	Reversal Sequence	Description
1	B_A_ ! _BA_	Singleton B at the beginning of the permutation is consecutive with a singleton A.
2	B_A _ ! _BA _	Singleton B at the beginning of the permutation is consecutive with the left endpoint A of a block *A* .
3	B_ A_ C_ ! A _B_ C_ ! _ AB_ C_ ! C _BA _ ! _ CBA _	Singleton B at the beginning of the permutation is consecutive with the last elements (A and C) of 2 separate blocks *A* and *C*.
4	B _A_ ! _ BA_	Left endpoint of block *B* at the beginning of the permutation is consecutive with a singleton A.
5	B _A _ ! _ BA _	Left endpoint of block *B* at the beginning of the permutation is consecutive with A in block *A* .
6	B C_D _ ! C B_D _ ! _B CD _	Right endpoint C in block *B* *C* at the beginning is consecutive with left endpoint D in block *D* .
7	B C_ D_ ! C B_D _ ! _ DC B_	Right endpoint C in block *B* *C* at the beginning is consecutive with right endpoint D in block *D*.
8	B C_ A_D_ ! D_A _C B_ ! _ A_D B_ ! B D_A _ ! _D A _	The block *B* *C* is at the beginning, left endpoint B is consecutive with right endpoint A in block *A*. The endpoint C of *B* *C* is consecutive with a singleton D occurring to the right of *A*.
9	B C_D_ A_ ! D_C B_ A_ ! D B_ A_ ! A _B D_ ! _ A D_	The block *B* *C* is at the beginning, left endpoint B is consecutive with right endpoint A in block *A*. The endpoint C of *B* *C* is consecutive with a singleton D occurring between the block *B* *C* and the block *A*.

intact until 2008, when a group of researchers at the University of Texas at Dallas lowered the bound to $\frac{18}{11}n \quad 1.636n$ with the use of high-powered

1. Reverse at $_{k+1}$.
   This results in the permutation: $_{k+1}$ $_{k+2}$

Algorithm 3. Suppose that $_c = (_1 \;\; _2 \cdots \; _k \;\; _{k+i} \;\; _{k+2} \cdots \; _{k+i-1} \;\; _{k+1} \;\; _{k+i+1} \cdots \; _n)$. The distance of the transposition, $(_{k+1} \;\; _{k+i})$ is greater than 2.

1. Reverse at $_k$.
   This results in the permutation: $_k \cdots \; _2 \;\; _1 \;\; _{k+i} \;\; _{k+2} \cdots \; _{k+i-1} \;\; _{k+1} \;\; _{k+i+1} \cdots \; _n$.
2. Reverse at $_{k+i}$.
   This results in the permutation: $_{k+i} \;\; _1 \cdots \; _k \;\; _{k+2} \cdots \; _{k+i-1} \;\; _{k+1} \;\; _{k+i+1} \cdots \; _n$.
3. Reverse at $_{k+1}$.
   This results in the permutation: $_{k+1} \;\; _{k+i-1} \cdots \; _{k+2} \;\; _k \cdots \; _1 \;\; _{k+i} \;\; _{k+i+1} \cdots \; _n$.
4. Reverse at $_{k+2}$.
   This results in the permutation: $_{k+2} \cdots \; _{k+i-1} \;\; _{k+1} \;\; _k \cdots \; _1 \;\; _{k+i} \;\; _{k+i+1} \cdots \; _n$.
5. Reverse at $_{k+i-1}$.
   This results in the permutation: $_{k+i-1} \cdots \; _{k+2} \;\; _{k+1} \;\; _k \cdots \; _1 \;\; _{k+i} \;\; _{k+i+1} \cdots \; _n$.
6. Reverse at $_1$.
   This results in the permutation: $_1 \;\; _2 \cdots \; _k \;\; _{k+1} \;\; _{k+2} \cdots \; _{k+i-1} \;\; _{k+i} \;\; _{k+i+1} \cdots \; _n$.
   This is the identity.

The general case of Algorithm 3 results in six pre x reversals. However, if the transposition is located at the beginning of the permutation, ie. $k = 0$, then steps 1 and step 2 are not necessary, and there are only four pre x reversals needed. Also, if the transposition is located at the second element of the permutation, ie. $k = 1$, then step 1 is not necessary, and there are only ve pre x reversals needed.

Lemma 5.8. *For $_c$ described above, the maximum number of reversals required to transform $_c$ to is 6.*

We combine the preceding three lemmas in the following theorem.

Theorem 5.9. *For $2 \; S_n$, such that can be decomposed into only one transposition, the maximum number of reversals required to transform to is 6.*

Example 5.10. Suppose we are given $2 \; S_8$ where $= (1\ 2\ 6\ 4\ 5\ 3\ 7\ 8)$. We see that the distance of the transposition, $(3; 6)$ is $6 \quad 3 = 3$.
Thus by Algorithm 3,

1. Reverse at 2: (2 1 6 4 5 3 7 8)
2. Reverse at 6: (6 1 2 4 5 3 7 8)
3. Reverse at 3: (3 5 4 2 1 6 7 8)
4. Reverse at 4: (4 5 3 2 1 6 7 8)
5. Reverse at 5: (5 4 3 2 1 6 7 8)
6. Reverse at 1: (1 2 3 4 5 6 7 8)

Thus, my algorithm only requires 6 reversals compared to Gates' algorithm, which requires 10 reversals.

As seen from the example above, my algorithm requires less reversals than Gates' algorithm. Gates' algorithm seems to require a maximum of 10 reversals as seen from the permutation below. We show the reversals for this particular permutation since the transposition has a large distance and is not located at the very beginning or end. We consider which types of permutations result in my algorithm requiring less reversals than Gates' algorithm in Section 6.

Lemma 5.11. *Given* $\;2\; S_n$ *such that* $= (_1 \; _2 \cdots \; _k \; _{k+i} \; _{k+2} \cdots \; _{k+i-1} \; _{k+1} \; _{k+i+1} \cdots \; _n)$. *By Gates' algorithm, falls into the case,* $B \quad C$

5. Reverse at $_{k+2}$: ($_{k+2} \ldots \; _{k+i-1} \; _{k+1} \; _{k} \ldots \; _{1} \; _{n} \ldots \; _{k+i+1} \; _{k+i}$)
   (Case 4)

6. Reverse at $_{k+i-1}$: ($_{k+i-1} \ldots \; _{k+2} \; _{k+1} \; _{k} \ldots \; _{1} \; _{n} \ldots \; _{k+i+1} \; _{k+i}$)
   (Case 5)

7. Reverse at $_{n}$: ($_{n} \; _{1} \ldots \; _{k} \; _{k+1} \; _{k+2} \ldots \; _{k+i-1} \; _{n-1} \ldots \; _{k+i+1} \; _{k+i}$)
   (Trivial algorithm)

8. Reverse at $_{k+i}$: ($_{k+i} \; _{k+i+1} \ldots \; _{n-1} \; _{k+i-1} \ldots \; _{k+2} \; _{k+1} \; _{k} \ldots \; _{1} \; _{n}$)
   (Trivial algorithm cont.)

9. Reverse at $_{n-1}$: ($_{n-1} \ldots \; _{k+i+1} \; _{k+i} \; _{k+i-1} \ldots \; _{k+2} \; _{k+1} \; _{k} \ldots \; _{1} \; _{n}$)
   (Trivial algorithm cont.)

10. Reverse at $_{1}$: ($_{1} \ldots \; _{k} \; _{k+1} \; _{k+2} \ldots \; _{k+i-1} \; _{k+i} \; _{K+i+1} \ldots \; _{n-1} \; _{n}$)
   (Trivial algorithm cont.)

This is the identity permutation. $\square$

We can also use these three algorithms when a permutation decomposes into two (or more) disjoe can 2(can)-334(2(can)4(2(ca(7.977utation.)]TJ/F389et2se2pping]TJ/F33

5. Reverse at 1: (1 2 3 4 5 8 7 6)

4. Reverse at $k+i+1$: $(k+i+1 \cdots k+j-1 \quad k+l \quad k+j+1 \cdots k+l-1 \quad k+j \quad k+l+1 \cdots n \quad 1 \cdots k-1 \quad k \quad k+i-1 \cdots k+1 \quad k+i)$
(Case 9 cont.)

5. Reverse at $k+1$: $(k+1 \cdots k+i-1 \quad k \quad k-1 \cdots 1 \quad n \cdots k+l+1 \quad k+j \quad k+l-1 \cdots k+j+1 \quad k+l \quad k+j-1 \cdots k+i+1 \quad k+i)$
(Case 4)

6. Reverse at $k+i-1$: $(k+i-1 \cdots k+1 \quad k \quad k-1 \cdots 1 \quad n \cdots k+l+1 \quad k+j \quad k+l-1 \cdots k+j+1 \quad k+l \quad k+j-1 \cdots k+i+1 \quad k+i)$
(Case 5)

7. Reverse at $k+l$: $(k+l \quad k+j+1 \cdots k+l-1 \quad k+j \quad k+l+1 \cdots n \quad 1 \cdots k-1 \quad k \quad k+1 \cdots k+i-l \quad k+j-1 \cdots k+i+1 \quad k+i)$
(Case 9)

8. Reverse at $k+j$: $(k+j \quad k+l-1 \cdots k+j+1 \quad k+l \quad k+l+1 \cdots n \quad 1 \cdots k-1 \quad k \quad k+1 \cdots k+i-l \quad k+j-1 \cdots k+i+1 \quad k+i)$
(Case 9 cont.)

9. Reverse at $k+i$: $(k+i \quad k+i+1 \cdots k+j-1 \quad k+i-1 \cdots k+1 \quad k \quad k-1 \cdots 1 \quad n \cdots k+l+1 \quad k+l \quad k+j+1 \cdots k+l-1 \quad k+j)$
(Case 9 cont.)

10. Reverse at $k+j-1$: $(k+j-1 \cdots k+i+1 \quad k+i \quad k+i-1 \cdots k+1 \quad k \quad k-1 \cdots 1 \quad n \cdots k+l+1 \quad k+l \quad k+j+1 \cdots k+l-1 \quad k+j)$
(Case 9 cont.)

11. Reverse at $k+l-1$: $(k+l-1 \cdots k+j+1 \quad k+l \quad k+l+1 \cdots n \quad 1 \cdots k-1 \quad k \quad k+1 \cdots k+i-1 \quad k+i \quad k+i+1 \cdots k+j-1 \quad k+j)$
(Case 4)

12. Reverse at $k+j+1$: $(k+j+1 \cdots k+l-1 \quad k+l \quad k+l+1 \cdots n \quad 1 \cdots k-1 \quad k \quad k+1 \cdots k+i-1 \quad k+i \quad k+i+1 \cdots k+j-1 \quad k+j)$
(Case 5)

13. Reverse at n: $(n \cdots k+l+1 \quad k+l \quad k+l-1 \cdots k+j+1 \quad 1 \cdots k-1 \quad k \quad k+1 \cdots k+i-1 \quad k+i \quad k+i+1 \cdots k+j-1 \quad k+j)$
(Trivial Algorithm)

14.

6 Statistics

We complete a similar analysis for the double, disjoint, non-overlapping transpositions. Let $x; y; a; b; c$ 2 Z such that, given 2 S_n,

$$(|_1 \ldots \quad _k$$

disjoint transpositions such that two transpositions are overlapping: ie. = (6 2 9 4 5 1 7 8 3).

non-disjoint transpositions, or a 3-cycle: ie. = (1 4 3 7 5 6 2).

8 References

[1] B. Chitturi, et al., An (18/11)*n* upper bound for sorting by pre x reversals, Theoretical Computer Science (2008), doi: 10.1016/j.tcs.2008.04.045.
[2] Gates W.H.; Papadimitriou, C.H. Bounds for sorting by pre x reversal. Discrete Math. 27 (1979), 47-57.

A Statistical Analysis

Single Transposition Cases

x	a	b		Alyssa	Gates	Di erence
1	0	0		1	1	0
1	0	1		1	1	0
1	0	2+		1	1	0
1	1	0		3	3	0
1	1	1		3	3	0
1	1	2+		3	7	4
1	2+	0		3	4	1
1	2+	1		3	3	0
1	2+	2+		3	7	4
2	0	0		1	1	0
2	0	1		1	1	0
2	0	2+		1	1	0
2	1	0		3	4	1
2	1	1		3	3	0
2	1	2+		3	7	4
2	2+	0		3	5	2
2	2+	1		3	3	0
2	2+	2+		3	7	4
3+	0	0		4	4	0
3+	0	1		4	8	4
3+	0	2+		4	8	4
3+	1	0		5	5	0
3+	1	1		5	8	3
3+	1	2+		5	5	0
3+	2+	0		6	6	0
3+	2+	1		6	10	4
3+	2+	2+		6	10	4

Double Disjoint Transposition Cases

Double Disjoint Transposition Cases (cont.)

x	y	a	b	c		Alyssa	Gates	Di erence
2	2	0	0	0		4	4	0
2	2	0	0	1		4	6	2
2	2	0	1	0		4	6	2
2	2	0	1	1		4	4	0
2	2	1	0	0		6	6	0
2	2	1	1	0		6	9	3
2	2	1	0	1		6	8	2
2	2	1	1	1		6	13	7
2	2	0	0	2+		4	6	2
2	2	0	2+	0		4	6	2
2	2	0	2+	2+		4	4	0
2	2	2+	0	0		6	6	0
2	2	2+	2+	0		6	8	2
2	2	2+	0	2+		6	8	2
2	2	2+	2+	2+		6	6	0
2	2	0	1	2+		4	4	0
2	2	0	2+	1		4	4	0
2	2	1	2+	0		6	8	2
2	2	2+	1	0		6	8	2
2	2	1	0	2+		6	8	2
2	2	2+	0	1		6	8	2
2	2	1	1	2+		6	11	5
2	2	1	2+	2+		6	10	4
2	2	1	2+	1		6	6	0
2	2	2+	1	1		6	6	0
2	2	2+	2+	1		6	6	0
2	2	2+	1	2+		6	6	0

Double Disjoint Transposition Cases (cont.)

x	y	a	b	c		Alyssa	Gates	Di erence
3+	3+	0	0	0		10	8	-2
3+	3+	0	0	1		10	12	2
3+	3+	0	1	0		10	10	0
3+	3+	0	1	1		10	14	4
3+	3+	1	0	0		11	14	3
3+	3+	1	1	0		11	11	0
3+	3+	1	0	1		11	10	3+ -1

3+

3+ 3+