The Pancake Problem: Prefix 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 differently-sized pancakes. Mathematically, this problem translates to flipping prefixes of permutations until the identity permutation is achieved. Bill Gates and Christos Papadimitriou created an algorithm in 1979 that improved the lower bound of the Pancake Problem. While Gates and Papadimitriou characterized a permutation based on blocks, I consider transposition decomposition and define a set of algorithms that require fewer reversals than Gates' algorithm in certain cases.

2 The Problem

In 1975, Jacob Goodman, under the name Harry Dweighter, proposed the Pancake Problem, which details a chef making pancakes in a busy diner. He makes a number of differently-sized pancakes and stacks them randomly on a plate. In order to arrange the pancakes in an aesthetically-pleasing manner, he flips portions of the stack so that the pancakes are ordered from the largest pancake on the bottom of the plate, the next largest on top of that, and so on, until the smallest pancake is on the top of the stack. The chef only has a spatula, so he can only place his spatula somewhere in the stack, lifting the portion above the spatula, flipping it, and then placing it back onto the stack. Goodman proposed this scenario wondering what the minimum number of flips the chef would need to perform in order to rearrange the stack of pancakes.

For example, consider the stack of pancakes below (Figure 1a). The chef can place the spatula below the third pancake (Figure 1b), flip it, and place it back on the stack to obtain a newly-arranged stack of pancakes (Figure 1c). He can then place the spatula below the fourth pancake (Figure 1d), flip it, and place it back on the stack to obtain another rearranged stack (Figure 1e). At this point, the stack is correctly rearranged.

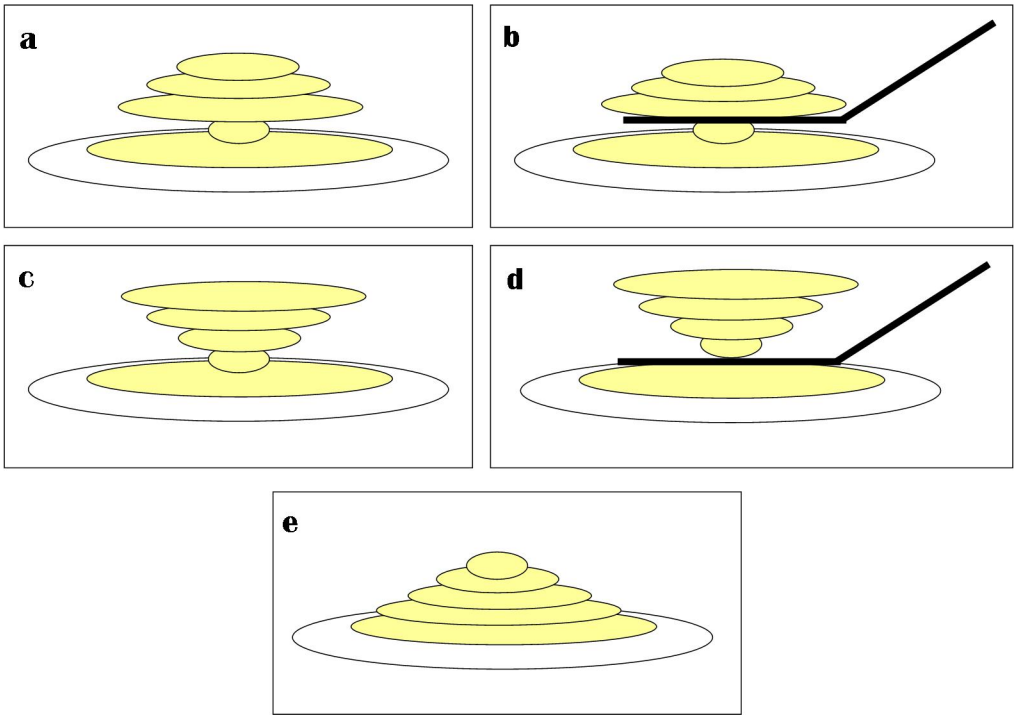

Figure 1: Flipping a Stack of Pancakes

We can translate this problem into the language of mathematics by rewriting a randomly arranged stack of n differently-sized pancakes as a permutation of n objects:

Definition 2.1. A **permutation** $\sigma \in S_n$ is an ordering of n distinct objects. S_n is the group of all permutations of n objects.

Example 2.2. Consider $\sigma \in S_5$. We can denote σ in three ways:

- Two line notation: $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 4 & 3 & 2 \end{pmatrix}$
- One line notation: (1 5 4 3 2) This is just the second line in the two line notation.
- Cyclic notation: (2 5)(3 4) This means 1 maps to 1, 2 maps to 5, 5 maps to 2, 3 maps to 4, and 4 maps to 3.

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 $\sigma \in S_n$, a **prefix reversal at** σ_i of $\sigma = (\sigma_1\ \sigma_2\ \ldots\ \sigma_i\ \ldots\ \sigma_n)$ is $\sigma' \in S_n$ such that $\sigma' = (\sigma_i\ \ldots\ \sigma_2\ \sigma_1\ \sigma_{i+1}\ \ldots\ \sigma_n)$.

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

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, $\sigma \in S_n$, to the identity is at most $2n$ reversals.*

Proof. We show this using the following algorithm:

1. Given $\sigma \in S_n$, reverse at the largest number that is not in its sorted position. (Note: a number is in its sorted position when $\sigma_i = i$.)
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 σ to its sorted position, it will take at most $2n$ reversals to transform σ to ι. □

Example 3.2. Given the permutation, $\sigma = (3\ 1\ 5\ 4\ 2)$. 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 σ to ι. 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 difficult 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 $\sigma \in S_n$. Gates and his advisor, Christos Papadimitriou, decreased the lower bound of reversals from $2n$ to $\frac{5n+5}{3} \approx 1.667n$, by classifying a permutation based on its block structure and creating an algorithm that will transform any $\sigma \in S_n$ to ι. What follows are a few definitions about his block structure.

Definition 4.1. Given the permutation, $\sigma \in S_n$.

- If $|\sigma_i - \sigma_j| \equiv 1 \ (mod\ n)$, then i is **consecutive** to j.
- If $|\sigma_i - \sigma_{i+1}| = 1$, then the pair $(i, i+1)$ is an **adjacency** in σ.
- A **block** is a maximal length sublist, $x = \sigma_i\ \sigma_{i+1} \ldots \sigma_{j-1}\ \sigma_j = y$, such that there is an adjacency between σ_a and σ_{a+1} for all $i \leq a \leq j$. We denote this block as $x \sim y$.
- The initial and final elements of a block are called the **endpoints** of the block.
- An element that does not occur in a block is called a **singleton**.

Example 4.2. Given $\sigma \in S_7$ such that $\sigma = (2\ 3\ 4\ 7\ 6\ 1\ 5)$,

- 1 is consecutive to 2, 2 is consecutive to 3, ..., 7 is consecutive to 8, and 8 is consecutive to 1.
- the pair $(7, 6)$ is an adjacency (and a block, since all adjacencies are blocks of size 2), and
- $(2\ 3\ 4)$ is a block in σ. The endpoints are 2 and 4.
- The elements 1 and 5 are singletons.

We can decompose a permutation based on the block structure of the permutation. For example, we will classify the permutation $\sigma \in S_7$ where $\sigma = (2\ 3\ 4\ 7\ 6\ 1\ 5)$. We analyze σ from the front of the permutation, looking for the elements consecutive to the first element of σ. We will always denote the first element of the permutation, B. In this case, $B = 2$, and σ begins with the block $(2\ 3\ 4)$, which we denote $B \sim C$. We then look for the element consecutive to the left endpoint of this block, which is 1. The element 1 is a singleton in σ since it is not part of any other block. We denote $A = 1$ in our classification, so that A is consecutive to $B = 2$. We do the same analysis for consecutive elements of the right endpoint of our initial block, which is 4. The only element consecutive to 4 is 5, since 3 is part of the block already. Since 5 is not part of any other block, it is a singleton, and we denote $D = 5$ in our classification, so that $C = 4$ is consecutive to $D = 5$. We denote any other elements between blocks and singletons by the symbol $\_$. We also separate the blocks and singletons marked by the symbol $\_$. Thus,

our classification of σ is $B \sim C\_ A \_ D$.

Gates and Papadimitriou thus define an algorithm which classifies 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 identified 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 $\sigma \in S_7$ where $\sigma = (2\ 3\ 4\ 7\ 6\ 1\ 5)$.

By Gate's Algorithm,

1. The permutation begins with the block $(2, 3, 4)$ and 2 is consecutive to 1 which is a singleton. Thus, by case 4, we reverse at 6: $(6\ 7\ 4\ 3\ 2\ 1\ 5)$.

2. The permutation begins with the block $(6, 7)$ and 6 is consecutive to 5 which is a singleton. Thus, by case 4, we reverse at 1: $(1\ 2\ 3\ 4\ 7\ 6\ 5)$.

3. The permutation begins with the block $(1, 2, 3, 4)$ and 1 is consecutive to 7, which is the left endpoint of the block $(7, 6, 5)$. Thus, by case 5, we reverse at 4: $(4\ 3\ 2\ 1\ 7\ 6\ 5)$.

4. The permutation is now a single block. At this point, the trivial algorithm is used to finish transforming the permutation. Thus, we reverse at 7: $(7\ 1\ 2\ 3\ 4\ 6\ 5)$.

5. Continuing with the trivial algorithm, we reverse at 5: $(5\ 6\ 4\ 3\ 2\ 1\ 7)$.

6. Continuing with the trivial algorithm, we reverse at 6: $(6\ 5\ 4\ 3\ 2\ 1\ 7)$.

7. Continuing with the trivial algorithm, we reverse at 1: $(1\ 2\ 3\ 4\ 5\ 6\ 7)$.

Thus, Gates' algorithm requires 7 reversals to transform σ to ι.

Using this algorithm, Gates and Papadimitriou showed that the lower bound for the minimum number of reversals needed to transform a permutation to the identity permutation is $\frac{5n+5}{3} \approx 1.667n$. This bound remained

Gates' Algorithm - Reversal Sequences		
Case	Reversal Sequence	Description
1	$B\_A\_$ $\rightarrow \_BA\_$	Singleton B at the beginning of the permutation is consecutive with a singleton A.
2	$B\_A \sim \_$ $\rightarrow \_BA \sim \_$	Singleton B at the beginning of the permutation is consecutive with the left endpoint A of a block $A \sim$.
3	$B\_ \sim A\_ \sim C\_$ $\rightarrow A \sim \_B\_ \sim C\_$ $\rightarrow \_ \sim AB\_ \sim C\_$ $\rightarrow C \sim \_BA \sim \_$ $\rightarrow \_ \sim CBA \sim \_$	Singleton B at the beginning of the permutation is consecutive with the last elements (A and C) of 2 separate blocks $\sim A$ and $\sim C$.
4	$B \sim \_A\_$ $\rightarrow \_ \sim BA\_$	Left endpoint of block $B \sim$ at the beginning of the permutation is consecutive with a singleton A.
5	$B \sim \_A \sim \_$ $\rightarrow \_ \sim BA \sim \_$	Left endpoint of block $B \sim$ at the beginning of the permutation is consecutive with A in block $A \sim$.
6	$B \sim C\_D \sim \_$ $\rightarrow C \sim B\_D \sim \_$ $\rightarrow \_B \sim CD \sim \_$	Right endpoint C in block $B \sim C$ at the beginning is consecutive with left endpoint D in block $D \sim$.
7	$B \sim C\_ \sim D\_$ $\rightarrow C \sim B\_D \sim \_$ $\rightarrow \_ \sim DC \sim B\_$	Right endpoint C in block $B \sim C$ at the beginning is consecutive with right endpoint D in block $\sim D$.
8	$B \sim C\_ \sim A\_D\_$ $\rightarrow D\_A \sim \_C \sim B\_$ $\rightarrow \_ \sim A\_D \sim B\_$ $\rightarrow B \sim D\_A \sim \_$ $\rightarrow \_D \sim A \sim \_$	The block $B \sim C$ is at the beginning, left endpoint B is consecutive with right endpoint A in block $\sim A$. The endpoint C of $B \sim C$ is consecutive with a singleton D occurring to the right of $\sim A$.
9	$B \sim C\_D\_ \sim A\_$ $\rightarrow D\_C \sim B\_ \sim A\_$ $\rightarrow D \sim B\_ \sim A\_$ $\rightarrow A \sim \_B \sim D\_$ $\rightarrow \_ \sim A \sim D\_$	The block $B \sim C$ is at the beginning, left endpoint B is consecutive with right endpoint A in block $\sim A$. The endpoint C of $B \sim C$ is consecutive with a singleton D occurring between the block $B \sim C$ and the block $\sim A$.

intact until 2008, when a group of researchers at the University of Texas at Dallas lowered the bound to $\frac{18}{11}n \approx 1.636n$ with the use of high-powered computers. This result would not have been achieved without Gates' large influence in the computing world, continuing his legacy in this problem.

5 My Algorithm

Instead of considering all permutations, I narrowed my research to include permutations that can be decomposed into only one or two disjoint transpositions.

Definition 5.1. Given $\sigma \in S_n$, a **transposition** is a mapping such that for $i, j \in \{1, 2, \ldots, n\}, \sigma_i = j, \sigma_j = i$ and $\sigma_k = k$ for all $k \neq i, j$.

Definition 5.2. A permutation $\sigma \in S_n$ is **disjoint and non-overlapping** if it is of the form, $\sigma = (\sigma_1\ \sigma_2 \ldots \sigma_{k-1}\ \sigma_{k+i}\ \sigma_{k+1} \ldots \sigma_{k+i-1}\ \sigma_k$ $\sigma_{k+i+1} \ldots \sigma_{k+j-1}\ \sigma_{k+l}\ \sigma_{k+j+1} \ldots \sigma_{k+l-1}\ \sigma_{k+j}\ \sigma_{k+l+1} \ldots \sigma_n)$.

Example 5.3. The transpositions, (1 2 5 4 3) and (1 4 3 2 5), are disjoint. However, the transpositions, (1 2 5 4 3) and (1 2 4 3 5), are not disjoint since 3 is mapped to both 5 and 4.

Definition 5.4. The **distance** of a transposition $(\sigma_1 \ldots \sigma_{i-1}\ \sigma_j\ \sigma_{i+1} \ldots \sigma_{j-1}$ $\sigma_i\ \sigma_{j+1} \ldots \sigma_n)$ is $|i - j|$.

Example 5.5. The permutation, $\sigma \in S_6$ where $\sigma = (1\ 2\ 6\ 4\ 5\ 3)$ is a transposition, since the 3 and 6 have been swapped. The distance of σ is $6 - 3 = 3$.

While Gates and Papadimitriou's algorithm is useful for all permutations, the algorithm is based on the structure of the permutation after every prefix reversal. When only considering permutations made up of one transposition, this algorithm can be simplified and we only need to consider three parameters: the distance $|\sigma_i - \sigma_j|$ of the transposition $(1 \ldots\ j \ldots\ i \ldots\ n)$, the number of elements before the transposition, and the number of elements after the transposition. By classifying permutations based on these three parameters, we can use three algorithms to transform a transposition to the identity using prefix reversals. These three algorithms are:

Algorithm 1. Suppose that $\sigma_a = (\sigma_1\ \sigma_2 \ldots \sigma_k\ \sigma_{k+2}\ \sigma_{k+1}\ \sigma_{k+3} \ldots \sigma_n)$. The distance of the transposition, $(\sigma_{k+1}\ \sigma_{k+2})$ is 1.

1. Reverse at σ_{k+1}.
   This results in the permutation: $\sigma_{k+1}\ \sigma_{k+2}\ \sigma_k \ldots \sigma_2\ \sigma_1\ \sigma_{k+3} \ldots \sigma_n$.

2. Reverse at σ_{k+2}.
   This results in the permutation: $\sigma_{k+2}\ \sigma_{k+1}\ \sigma_k \ldots \sigma_2\ \sigma_1\ \sigma_{k+3} \ldots \sigma_n$.

3. Reverse at σ_1.
   This results in the permutation: $\sigma_1\ \sigma_2 \ldots \sigma_k\ \sigma_{k+1}\ \sigma_{k+2}\ \sigma_{k+3} \ldots \sigma_n$.
   This is the identity.

The general case of Algorithm 1 results in three prefix reversals. However, if the transposition is located at the beginning of the permutation, ie. $k = 0$, then step 1 and step 2 are not necessary. Thus, there is only one prefix reversal needed. Thus,

Lemma 5.6. *For σ_a described above, the maximum number of reversals required to transform σ_a to ι is 3.*

Algorithm 2. Suppose that $\sigma_b = (\sigma_1\ \sigma_2 \ldots \sigma_k\ \sigma_{k+3}\ \sigma_{k+2}\ \sigma_{k+1}\ \sigma_{k+4} \ldots \sigma_n)$. The distance of the transposition, $(\sigma_{k+1}\ \sigma_{k+3})$ is 2.

1. Reverse at σ_{k+1}.
   This results in the permutation: $\sigma_{k+1}\ \sigma_{k+2}\ \sigma_{k+3}\ \sigma_k \ldots \sigma_2\ \sigma_1\ \sigma_{k+4} \ldots \sigma_n$.

2. Reverse at σ_{k+3}.
   This results in the permutation: $\sigma_{k+3}\ \sigma_{k+2}\ \sigma_{k+1}\ \sigma_k \ldots \sigma_2\ \sigma_1\ \sigma_{k+4} \ldots \sigma_n$.

3. Reverse at σ_1.
   This results in the permutation: $\sigma_1\ \sigma_2 \ldots \sigma_k\ \sigma_{k+1}\ \sigma_{k+2}\ \sigma_{k+3}\ \sigma_{k+4} \ldots \sigma_n$.
   This is the identity.

Similar to the first algorithm,the general case of Algorithm 2 results in three prefix reversals. However, if the transposition is located at the beginning of the permutation, ie. $k = 0$, then step 1 and step 2 are not necessary, and there is only one prefix reversal needed. Thus,

Lemma 5.7. *For σ_b described above, the maximum number of reversals required to transform σ_b to ι is 3.*

Algorithm 3. Suppose that $\sigma_c = (\sigma_1\ \sigma_2 \ldots \sigma_k\ \sigma_{k+i}\ \sigma_{k+2} \ldots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_{k+i+1} \ldots \sigma_n)$. The distance of the transposition, $(\sigma_{k+1}\ \sigma_{k+i})$ is greater than 2.

1. Reverse at σ_k.
   This results in the permutation: $\sigma_k \ldots \sigma_2\ \sigma_1\ \sigma_{k+i}\ \sigma_{k+2} \ldots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_{k+i+1} \ldots \sigma_n$.

2. Reverse at σ_{k+i}.
   This results in the permutation: $\sigma_{k+i}\ \sigma_1 \ldots \sigma_k\ \sigma_{k+2} \ldots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_{k+i+1} \ldots \sigma_n$.

3. Reverse at σ_{k+1}.
   This results in the permutation: $\sigma_{k+1}\ \sigma_{k+i-1} \ldots \sigma_{k+2}\ \sigma_k \ldots \sigma_1\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_n$.

4. Reverse at σ_{k+2}.
   This results in the permutation: $\sigma_{k+2} \ldots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_k \ldots \sigma_1\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_n$.

5. Reverse at σ_{k+i-1}.
   This results in the permutation: $\sigma_{k+i-1} \ldots \sigma_{k+2}\ \sigma_{k+1}\ \sigma_k \ldots \sigma_1\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_n$.

6. Reverse at σ_1.
   This results in the permutation: $\sigma_1\ \sigma_2 \ldots \sigma_k\ \sigma_{k+1}\ \sigma_{k+2} \ldots \sigma_{k+i-1}\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_n$.
   This is the identity.

The general case of Algorithm 3 results in six prefix 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 prefix 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 five prefix reversals needed.

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

We combine the preceding three lemmas in the following theorem.

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

Example 5.10. Suppose we are given $\sigma \in S_8$ where $\sigma = (1\ 2\ 6\ 4\ 5\ 3\ 7\ 8)$. We see that the distance of the transposition, $(3, 6)$ is $6 - 3 = 3$.
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 $\sigma \in S_n$ such that*
$\sigma = (\sigma_1\ \sigma_2 \dots \sigma_k\ \sigma_{k+i}\ \sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_{k+i+1} \dots \sigma_n)$. By Gates' algorithm, σ falls into the case, $B \sim C \_ D \_ \sim A \_$, and thus requires 10 reversals to obtain the identity permutation.

Proof. Given $\sigma = (\sigma_1\ \sigma_2 \dots \sigma_k\ \sigma_{k+i}\ \sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_{k+i+1} \dots \sigma_n)$.

1. Reverse at σ_{k+1}: $(\sigma_{k+1}\ \sigma_{k+i-1} \dots \sigma_{k+2}\ \sigma_{k+i}\ \sigma_k \dots \sigma_1\ \sigma_{k+i+1} \dots \sigma_n)$ (Case 9)
2. Reverse at σ_{k+i}: $(\sigma_{k+i}\ \sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_k \dots \sigma_1\ \sigma_{k+i+1} \dots \sigma_n)$ (Case 9 cont.)
3. Reverse at σ_n: $(\sigma_n \dots \sigma_{k+i+1}\ \sigma_1 \dots \sigma_k\ \sigma_{k+1}\ \sigma_{k+i-1} \dots \sigma_{k+2}\ \sigma_{k+i})$ (Case 9 cont.)
4. Reverse at σ_{k+i+1}: $(\sigma_{k+i+1} \dots \sigma_n\ \sigma_1 \dots \sigma_k\ \sigma_{k+1}\ \sigma_{k+i-1} \dots \sigma_{k+2}\ \sigma_{k+i})$ (Case 9 cont.)

5. Reverse at σ_{k+2}: $(\sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{k+1}\ \sigma_k \dots \sigma_1\ \sigma_n \dots \sigma_{k+i+1}\ \sigma_{k+i})$ (Case 4)

6. Reverse at σ_{k+i-1}: $(\sigma_{k+i-1} \dots \sigma_{k+2}\ \sigma_{k+1}\ \sigma_k \dots \sigma_1\ \sigma_n \dots \sigma_{k+i+1}\ \sigma_{k+i})$ (Case 5)

7. Reverse at σ_n: $(\sigma_n\ \sigma_1 \dots \sigma_k\ \sigma_{k+1}\ \sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{n-1} \dots \sigma_{k+i+1}\ \sigma_{k+i})$ (Trivial algorithm)

8. Reverse at σ_{k+i}: $(\sigma_{k+i}\ \sigma_{k+i+1} \dots \sigma_{n-1}\ \sigma_{k+i-1} \dots \sigma_{k+2}\ \sigma_{k+1}\ \sigma_k \dots \sigma_1\ \sigma_n)$ (Trivial algorithm cont.)

9. Reverse at σ_{n-1}: $(\sigma_{n-1} \dots \sigma_{k+i+1}\ \sigma_{k+i}\ \sigma_{k+i-1} \dots \sigma_{k+2}\ \sigma_{k+1}\ \sigma_k \dots \sigma_1\ \sigma_n)$ (Trivial algorithm cont.)

10. Reverse at σ_1: $(\sigma_1 \dots \sigma_k\ \sigma_{k+1}\ \sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{k+i}\ \sigma_{K+i+1} \dots \sigma_{n-1}\ \sigma_n)$ (Trivial algorithm cont.)

This is the identity permutation. □

We can also use these three algorithms when a permutation decomposes into two (or more) disjoint, non-overlapping transpositions.

Theorem 5.12. *For $\sigma \in S_n$ where $\sigma = (\sigma_1 \dots \sigma_k\ \sigma_{k+i} \dots \sigma_{k+1}\ \sigma_{k+i+1} \dots \sigma_{k+j-1}\ \sigma_{k+l} \dots \sigma_{k+j}\ \sigma_{k+l+1} \dots \sigma_n)$ (ie. where σ can be decomposed into two disjoint, non-overlapping transpositions), the maximum number of reversals required to transform σ to ι is 12 when using my algorithms stated above.*

We see this with the following example:

Example 5.13. Suppose we are given $\sigma \in S_8$ where $\sigma = (1\ 5\ 3\ 4\ 2\ 8\ 7\ 6)$. This can be decomposed into two transpositions: $(5, 2)$ (distance 3) and $(8, 6)$ (distance 2).
First, by Algorithm 3,

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

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

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

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

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

Then, by Algorithm 2,

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

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

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

Thus, my algorithm only requires 8 reversals compared to Gates' Algorithm, which requires 9 reversals.

As before, we can see from the three algorithms that the maximum number of reversals required to transform a permutation two disjoint, non-overlapping transpositions to the identity permutation is 12 reversals. We show that the number of reversals required for this type of permutation, where both transpositions have large distances and neither are located at the beginning, end or right next to each other, for Gates' algorithm is 15 reversals:

Lemma 5.14. *Given* $\sigma \in S_n$ *such that* $\sigma = (\sigma_1\ \sigma_2 \dots \sigma_{k-1}\ \sigma_{k+i}\ \sigma_{k+1} \dots \sigma_{k+i-1}\ \sigma_k\ \sigma_{k+i+1} \dots \sigma_{k+j-1}\ \sigma_{k+l}\ \sigma_{k+j+1} \dots \sigma_{k+l-1}\ \sigma_{k+j}\ \sigma_{k+l+1} \dots \sigma_n)$. *By Gates' algorithm,* σ *falls into the case,* $B \sim C\ \_\ D\ \_ \sim A\ \_$, *and thus requires 15 reversals to obtain the identity permutation.*

Proof. Given $\sigma = (\sigma_1\ \sigma_2 \dots \sigma_{k-1}\ \sigma_{k+i}\ \sigma_{k+1} \dots \sigma_{k+i-1}\ \sigma_k\ \sigma_{k+i+1} \dots \sigma_{k+j-1}\ \sigma_{k+l}\ \sigma_{k+j+1} \dots \sigma_{k+l-1}\ \sigma_{k+j}\ \sigma_{k+l+1} \dots \sigma_n)$.

1. Reverse at σ_k: $(\sigma_k\ \sigma_{k+i-1} \dots \sigma_{k+1}\ \sigma_{k+i}\ \sigma_{k-1} \dots \sigma_1\ \sigma_{k+i+1} \dots \sigma_{k+j-1}\ \sigma_{k+l}\ \sigma_{k+j+1} \dots \sigma_{k+l-1}\ \sigma_{k+j}\ \sigma_{k+l+1} \dots \sigma_n)$
   (Case 9)

2. Reverse at σ_{k+i}: $(\sigma_{k+i}\ \sigma_{k+1} \dots \sigma_{k+i-1}\ \sigma_k\ \sigma_{k-1} \dots \sigma_1\ \sigma_{k+i+1} \dots \sigma_{k+j-1}\ \sigma_{k+l}\ \sigma_{k+j+1} \dots \sigma_{k+l-1}\ \sigma_{k+j}\ \sigma_{k+l+1} \dots \sigma_n)$
   (Case 9 cont.)

3. Reverse at σ_n: $(\sigma_n \dots \sigma_{k+l+1}\ \sigma_{k+j}\ \sigma_{k+l-1} \dots \sigma_{k+j+1}\ \sigma_{k+l}\ \sigma_{k+j-1} \dots \sigma_{k+i+1}\ \sigma_1 \dots \sigma_{k-1}\ \sigma_k\ \sigma_{k+i-1} \dots \sigma_{k+1}\ \sigma_{k+i})$
   (Case 9 cont.)

4. Reverse at σ_{k+i+1}: $(\sigma_{k+i+1} \ldots \sigma_{k+j-1}\ \sigma_{k+l}\ \sigma_{k+j+1} \ldots \sigma_{k+l-1}$ $\sigma_{k+j}\ \sigma_{k+l+1} \ldots \sigma_n\ \sigma_1 \ldots \sigma_{k-1}\ \sigma_k\ \sigma_{k+i-1} \ldots \sigma_{k+1}\ \sigma_{k+i})$
(Case 9 cont.)

5. Reverse at σ_{k+1}: $(\sigma_{k+1} \ldots \sigma_{k+i-1}\ \sigma_k\ \sigma_{k-1} \ldots \sigma_1\ \sigma_n \ldots \sigma_{k+l+1}$ $\sigma_{k+j}\ \sigma_{k+l-1} \ldots \sigma_{k+j+1}\ \sigma_{k+l}\ \sigma_{k+j-1} \ldots \sigma_{k+i+1}\ \sigma_{k+i})$
(Case 4)

6. Reverse at σ_{k+i-1}: $(\sigma_{k+i-1} \ldots \sigma_{k+1}\ \sigma_k\ \sigma_{k-1} \ldots \sigma_1\ \sigma_n \ldots \sigma_{k+l+1}$ $\sigma_{k+j}\ \sigma_{k+l-1} \ldots \sigma_{k+j+1}\ \sigma_{k+l}\ \sigma_{k+j-1} \ldots \sigma_{k+i+1}\ \sigma_{k+i})$
(Case 5)

7. Reverse at σ_{k+l}: $(\sigma_{k+l}\ \sigma_{k+j+1} \ldots \sigma_{k+l-1}\ \sigma_{k+j}\ \sigma_{k+l+1} \ldots \sigma_n$ $\sigma_1 \ldots \sigma_{k-1}\ \sigma_k\ \sigma_{k+1} \ldots \sigma_{k+i-l}\ \sigma_{k+j-1} \ldots \sigma_{k+i+1}\ \sigma_{k+i})$
(Case 9)

8. Reverse at σ_{k+j}: $(\sigma_{k+j}\ \sigma_{k+l-1} \ldots \sigma_{k+j+1}\ \sigma_{k+l}\ \sigma_{k+l+1} \ldots \sigma_n\ \sigma_1 \ldots \sigma_{k-1}$ $\sigma_k\ \sigma_{k+1} \ldots \sigma_{k+i-l}\ \sigma_{k+j-1} \ldots \sigma_{k+i+1}\ \sigma_{k+i})$
(Case 9 cont.)

9. Reverse at σ_{k+i}: $(\sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_{k+j-1}\ \sigma_{k+i-1} \ldots \sigma_{k+1}\ \sigma_k\ \sigma_{k-1} \ldots \sigma_1$ $\sigma_n \ldots \sigma_{k+l+1}\ \sigma_{k+l}\ \sigma_{k+j+1} \ldots \sigma_{k+l-1}\ \sigma_{k+j})$
(Case 9 cont.)

10. Reverse at σ_{k+j-1}: $(\sigma_{k+j-1} \ldots \sigma_{k+i+1}\ \sigma_{k+i}\ \sigma_{k+i-1} \ldots \sigma_{k+1}\ \sigma_k\ \sigma_{k-1} \ldots \sigma_1$ $\sigma_n \ldots \sigma_{k+l+1}\ \sigma_{k+l}\ \sigma_{k+j+1} \ldots \sigma_{k+l-1}\ \sigma_{k+j})$
(Case 9 cont.)

11. Reverse at σ_{k+l-1}: $(\sigma_{k+l-1} \ldots \sigma_{k+j+1}\ \sigma_{k+l}\ \sigma_{k+l+1} \ldots \sigma_n\ \sigma_1 \ldots \sigma_{k-1}$ $\sigma_k\ \sigma_{k+1} \ldots \sigma_{k+i-1}\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_{k+j-1}\ \sigma_{k+j})$
(Case 4)

12. Reverse at σ_{k+j+1}: $(\sigma_{k+j+1} \ldots \sigma_{k+l-1}\ \sigma_{k+l}\ \sigma_{k+l+1} \ldots \sigma_n\ \sigma_1 \ldots \sigma_{k-1}$ $\sigma_k\ \sigma_{k+1} \ldots \sigma_{k+i-1}\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_{k+j-1}\ \sigma_{k+j})$
(Case 5)

13. Reverse at σ_n: $(\sigma_n \ldots \sigma_{k+l+1}\ \sigma_{k+l}\ \sigma_{k+l-1} \ldots \sigma_{k+j+1}\ \sigma_1 \ldots \sigma_{k-1}\ \sigma_k$ $\sigma_{k+1} \ldots \sigma_{k+i-1}\ \sigma_{k+i}\ \sigma_{k+i+1} \ldots \sigma_{k+j-1}\ \sigma_{k+j})$
(Trivial Algorithm)

14. Reverse at σ_{k+j}: $(\sigma_{k+j}\ \sigma_{k+j-1} \dots \sigma_{k+i+1}\ \sigma_{k+i}\ \sigma_{k+i-1} \dots \sigma_{k+1}\ \sigma_k$ $\sigma_{k-1} \dots \sigma_1\ \sigma_{k+j+1} \dots \sigma_{k+l-1}\ \sigma_{k+l}\ \sigma_{k+l+1} \dots \sigma_n)$
(Trivial Algorithm cont.)

15. Reverse at σ_1: $(\sigma_1 \dots \sigma_{k-1}\ \sigma_k\ \sigma_{k+1} \dots \sigma_{k+i-1}\ \sigma_{k+i}\ \sigma_{k+i+1} \dots \sigma_{k+j-1}$ $\sigma_{k+j}\ \sigma_{k+j+1} \dots \sigma_{k+l-1}\ \sigma_{k+l}\ \sigma_{k+l+1} \dots \sigma_n)$
(Trivial Algorithm cont.)

This is the identity permutation. □

It seems plausible that for single and disjoint double transpositions, my algorithm will require fewer reversals than Gates' algorithm since the maximum number of reversals needed for my algorithm is 12, while the maximum number of reversals needed for Gates' algorithm is 15.

However, for some permutations which can be decomposed into two disjoint, non-overlapping transpositions, Gates' algorithm does require fewer reversals than my algorithm. We see this in the following example,

Example 5.15. Suppose we are given $\sigma \in S_7$ where $\sigma = (1\ 2\ 3\ 5\ 4\ 7\ 6)$. This can be decomposed into two transpositions: $(4, 5)$ (distance 1) and $(6, 7)$ (distance 1). We will transform σ to ι using my algorithm.
First, by Algorithm 1,

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

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

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

Then, by Algorithm 1,

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

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

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

Thus, my algorithm requires 6 reversals compared to Gates' Algorithm, which only requires 5 reversals.

6 Statistics

After creating the three algorithms for single and disjoint, non-overlapping double transpositions, we now investigate what causes the most significant difference between the number of reversals required for Gates' algorithm and the number of reversals required for my algorithm.

For a single transposition, we can define variables $a, b, x \in \mathbb{Z}$ that characterizes the location and distance of the transposition:

Given $\sigma \in S_n$,

$$\sigma = (\underbrace{\sigma_1\ \sigma_2 \dots \sigma_k}_{a} \underbrace{\sigma_{k+i}\ \sigma_{k+2} \dots \sigma_{k+i-1}\ \sigma_{k+1}}_{x} \underbrace{\sigma_{k+i+1} \dots \sigma_n}_{b}).$$

When defining my algorithms, we saw that the distance of the transposition, x, can either be 1, 2, or ≥ 3. Also, we saw that a and b can either be 0, 1, or ≥ 2. Thus, there are three possible values for x, three possible values for a, and three possible values for b, resulting in $3^3 = 27$ possible cases for a single transposition.

Theorem 6.1. *My algorithm requires fewer reversals than Gates' algorithm for 12 cases, and the same number of reversals for 15 cases.*

Therefore, my algorithm shows 100 percent improvement for the single transposition case. We define improvement as requiring either the same number or fewer reversals than Gates' algorithm. (See the Appendix for the table of cases and their respective differences in the number of reversals.)

In an attempt to predict the difference in the number of reversals required by Gates' algorithm and the number of reversals required by my algorithm (a positive difference means that my algorithm is more efficient), I used Minitab to compute a regression model:

$$\text{Difference} = -0.833 + 0.556x + 0.611a + 1.11b$$

Although none of the coefficients of the variables are very large – making the model statistically insignificant – all the coefficients are greater than 0. This means that the difference will tend to be positive and, thus, my algorithm shows improvement compared to Gates' algorithm.

We complete a similar analysis for the double, disjoint, non-overlapping transpositions. Let $x, y, a, b, c \in \mathbb{Z}$ such that, given $\sigma \in S_n$,

$$\big(\underbrace{\sigma_1 \dots \sigma_k}_{a}\ \underbrace{\sigma_{k+i} \dots \sigma_{k+1}}_{x}\ \underbrace{\sigma_{k+i+1} \dots \sigma_{k+j-1}}_{b}\ \underbrace{\sigma_{k+l} \dots \sigma_{k+j}}_{y}\ \underbrace{\sigma_{k+l+1} \dots \sigma_n}_{c}\big).$$

Once again, x and y can either be 1, 2, or ≥ 3, and a, b, and c can either be 0, 1, ≥ 2. I only considered cases where $x = y$, so there are $3^4 = 81$ cases for a double transposition. For each case, I found the difference between the number of reversals for Gates' algorithm and the number of reversals for my algorithm (see the Appendix).

Theorem 6.2. *My algorithm requires fewer reversals than Gates' algorithm for 43 cases, it requires the same number of reversals for 30 cases, and requires more reversals for 8 cases.*

Thus, my algorithm shows a 90.12 percent improvement compared to Gates' algorithm.

Predicting the difference using a regression model, we see that

$$\text{Difference} = 0.407 - 0.167a + 0.148x + 0.148b + 0.611c$$

Like in the single transposition case, we find that the model is not statistically significant, since the coefficients are less than 1. However, we see that the coefficient for a is negative, which leads us to hypothesize that the location of the first transposition could affect the difference.

7 Conclusion

For the single transposition and double, disjoint, non-overlapping transpositions, we see that my algorithm requires less reversals than Gates' algorithm for most permutations. I hope to continue to analyze the type of permutation and its effects on the difference between the number of reversals without proceeding case-by-case. I also would like to extend my analysis to other types of double transpositions, including

- disjoint transpositions such that one transposition is nested within the other: ie. $\sigma = (7\ 4\ 3\ 2\ 5\ 6\ 1)$.

- disjoint transpositions such that two transpositions are overlapping: ie. $\sigma = (6\ 2\ 9\ 4\ 5\ 1\ 7\ 8\ 3)$.
- non-disjoint transpositions, or a 3-cycle: ie. $\sigma = (1\ 4\ 3\ 7\ 5\ 6\ 2)$.

8 References

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

A Statistical Analysis

Single Transposition Cases

x	a	b		Alyssa	Gates	Difference
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

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

Double Disjoint Transposition Cases (cont.)

x	y	a	b	c		Alyssa	Gates	Difference
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	Difference
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	-1
3+	3+	1	1	1		11	11	0
3+	3+	0	0	2+		10	12	2
3+	3+	0	2+	0		10	10	0
3+	3+	0	2+	2+		10	15	5
3+	3+	2+	0	0		12	10	-2
3+	3+	2+	2+	0		12	13	1
3+	3+	2+	0	2+		12	14	2
3+	3+	2+	2+	2+		12	15	3
3+	3+	0	1	2+		10	15	5
3+	3+	0	2+	1		10	14	4
3+	3+	1	2+	0		11	11	0
3+	3+	2+	1	0		12	12	0
3+	3+	1	0	2+		11	10	-1
3+	3+	2+	0	1		12	10	-2
3+	3+	1	1	2+		11	11	0
3+	3+	1	2+	2+		11	15	4
3+	3+	1	2+	1		11	11	0
3+	3+	2+	1	1		12	15	3
3+	3+	2+	2+	1		12	15	3
3+	3+	2+	1	2+		12	15	3