A New Upper Bound for Sorting Permutations with Prefix Transpositions

1Pramod Nair, 1Rajan Sundararavardhan, 2Bhadrachalam Chitturu
1Dept of Mathematics, ASAS, AVVP, Amritapuri Campus, India
2Dept of Computer Science, University of Texas at Dallas, Richardson Texas, USA.

Abstract. Permutations are discrete structures that naturally model a genome where every gene occurs exactly once. Sequences in a permutation over the given alphabet \( \Sigma \), each symbol of \( \Sigma \) appears exactly once. A transposition operation on a given permutation \( \pi \) exchanges two adjacent sublists of \( \pi \). If one of these sublists is restricted to be a prefix then one obtains a prefix transposition. The symmetric group of permutations with \( n \) symbols derived from the alphabet \( \Sigma = \{0, 1, \ldots, n-1\} \) is denoted by \( S_n \). The symmetric prefix transposition distance between \( \pi^* \in S_n \) and \( \pi^* \in S_n \) is the minimum number of prefix transpositions that are needed to transform \( \pi^* \) into \( \pi^* \). It is known that transforming an arbitrary \( \pi^* \in S_n \) into an arbitrary \( \pi^* \in S_n \) is equivalent to sorting some \( \pi^* \in S_n \). Thus, upper bound for transforming any \( \pi^* \in S_n \) into any \( \pi^* \in S_n \) with prefix transpositions is simply the upper bound to sort any permutation \( \pi \in S_n \). The current upper bound is \( n-\log_2 n \) for prefix transposition distance over \( S_n \). In this article, we improve the same to \( n-\log_3 n \).

Keywords: Cayley graphs, permutations, sorting, prefix transpositions, diameter, upper bound.

1 Introduction

A permutation over the given alphabet \( \Sigma = \{0, 1, \ldots, n-1\} \) is an instance of a bijection from \( \Sigma \) to itself. All such bijections form the symmetric group \( S_n \). A transposition exchanges two adjacent sublists in a permutation. Given the alphabet \{ “eat”, “play”, “rest”\}, the sequence “eat play rest” becomes “eat rest play” by applying a transposition. Prefix transposition operation is a restricted version of transposition operation where one of the moved sublist is a prefix of the permutation. In the example shown above an application of a prefix transposition would yield “play eat rest”. We enclose the moved sublist in parentheses and the destination position is marked with an asterisk [C15]. The remainder of the article closely adopts the terminology of [C15].

Prefix transposition operation on a sequence of length \( n \) has \( n(n-1)/2 \) generators. For \( S_3 \) the generator set for prefix transpositions is \{[1 0 2], [1 2 0], [2 0 1] \}. \( I_n=\{0, 1, \ldots, n-1\} \in S_n \) is the identity (sorted) permutation. Transforming any permutation \( \pi^* \in S_n \) into any permutation \( \pi^* \in S_n \) is equivalent to sorting some \( \pi^* \in S_n \) i.e. transforming \( \pi^* \) to \( I_n \), here \( \pi^* \) is applied to the inverse of \( \pi^* \). Thus, in this article only sorting is considered. Operations mimic genomic mutations and permutations mimic genes [FL09]. A Cayley graph models an interconnection networks corresponding to an operation. Here a vertex is labelled by a permutation and an edge \((u,v)\) indicates that a generator \( g \) exists in the generator set of the operation such that \( ug = v \) [AK89, LJD93].

Given a permutation \( \pi = (\pi_1, \pi_2, \ldots, \pi_n) \), a transposition \( \delta(i, j+1, k) \) moves the sublist \((\pi_i, \ldots, \pi_k)\) into the position immediately following \( \pi_i \). Transposition operation on permutations has been studied extensively, e.g. [BF98, BF12]. Several variations of this operation have been studied. Heath and Vergara study sorting permutations by bounded block moves [HV98] whereas Feng et al. study sorting permutations with cyclic adjacent transpositions [FC10]. A prefix transposition, a move henceforth, needs only to describe the position immediately succeeding the prefix and the destination position. The prefix transposition \( \rho(i+1, j) \) on \( \pi = (\pi_1, \pi_2, \ldots, \pi_n) \) positions the prefix \((\pi_i, \ldots, \pi_j)\) immediately after \( \pi_i \). \( \alpha \rightarrow \beta \) denotes that a single move transforms \( \alpha \) into \( \beta \). \( \rho(j-i+1, j) \) is the inverse move of \( \rho(i+1, j) \); i.e. executing \( \rho(i+1, j) \) followed by \( \rho(j-i+1, j) \) on \( \pi \) yields \( \pi \). Given permutations \( \alpha, \beta \in S_n \), the symmetric prefix transposition distance, \( d_\rho(\alpha, \beta) \), between \( \alpha \) and \( \beta \), is the minimum number of moves that transforms \( \alpha \) into \( \beta \). The prefix transposition diameter over \( S_n \), denoted by \( d_\rho(S_n) \) equals \( \max_{\alpha, \beta \in S_n} d_\rho(\alpha, \beta) \) which equals \( \max_{\alpha, \beta \in S_n} d_\rho(\alpha, I_n) \). We establish a new upper bound for \( d_\rho(S_n) \), denoted by \( \Phi(S_n) \), by establishing an upper bound for \( \max_{\alpha, \beta \in S_n} d_\rho(\alpha, I_n) \).

The articles [CS08, CS12, DM02, C07] study sorting permutations with prefix transpositions whereas [C07, CS08] studied prefix transpositions on strings. Given \( \pi \in S_n \), if \( \pi_{i+1} = \pi_i + 1 \) then \( \pi_i \) and \( \pi_{i+1} \) form an adjacency. \( I_n \) has exactly \( n-1 \) adjacencies since \( \forall i, \pi_i = k-1 \). The reverse order permutation where \( \forall i, \pi_i = n-k \), denoted by \( R_n \), has zero adjacencies. Thus, a natural way an algorithm tries to sort a permutation is by creating new adjacencies.
Consider the permutation \( \pi = 6 4 3 0 1 2 5 \in \Sigma \) over \( \Sigma = \{0, 1, 2, 3, 4, 5, 6\} \). Since, we do not break any adjacencies the block 0 1 2 will stay together. Thus, we reduce \( \pi \) into \( \pi^* = 4 2 1 0 3 \) where 0 1 2 is replaced by 0 and the values of all the elements with value greater than two are decremented by two (the number of adjacencies in the block 0 1 2). This yields a permutation in \( \Sigma_1 \) which denotes a permutation in \( \Sigma \). This process is repeated until all the adjacencies are eliminated, so, 4 2 1 0 3 is in fact reduced or it is irreducible. Given a permutation \( \pi \) and the corresponding reduced permutation \( \pi^* \), Christie [C99] showed that \( d(\pi, I_0) = d(\pi^*, I_0) \) where \( d(\alpha, \beta) \) is the transposition distance between \( \alpha \) and \( \beta \); Christie also showed that there is an optimal sequence of transpositions that does not break any adjacencies. It follows that \( d(\pi, I_0) = d(\pi^*, I_0) \). In order to establish an upper bound to sort any permutation in \( \Sigma_0 \) we need only consider irreducible permutations that correspond to the worst case [CS12]. A move can either create or destroy zero, one, or two adjacencies. In this article we avoid moves that break existing adjacencies. A move that does not create (or destroy) any adjacency is called a blank, moves that create one or two adjacencies each are called a single and a double respectively [CS12]. The distance between two symbols \( x, y \) in the alphabet is defined as \( (x - y) \mod n \); this measure is asymmetric. It corresponds to the number of positions to be traversed in the counter-clockwise direction over the cyclic identity permutation to reach position \( y \) from position \( x \) [CS12]. For example distance(5, 2) in \( \Sigma_1 \) equals three whereas distance(2, 5) equals five. This is easy to visualize if one imagines that 0 is adjacent to 7 in \( I_0 = (0 1 2 3 4 5 6 7) \). So, distance(2, 5) indicates the ordered traversal: (1 0 7 6 5).

A greedy method was employed to show an upper bound of \( n - \log_2 n \) [CS08]. In the process of sorting \( \pi \), a symbol that appears in the first position of \( \pi \) or any subsequent permutation is called a visited symbol otherwise, it is an unvisited symbol [CS08]. Further, a visited symbol appears in the first position of a permutation exactly once [CS08] in the sequence of permutations that starts with the given permutation \( \pi \) and ends with \( I_0 \). In [CS08] it is shown that any permutation \( \pi \) can be written as \( \pi = (t_1, …, s_i, t_2, …, s_j, …, t_l) \) where \( t_1 \) is a visited symbol and \( i, j \) is an unvisited symbol immediately preceding \( t \) (an \( s \) need not exist but a \( t \) is mandatory). This method was later improved to show a better upper bound of \( n - \log_2 n \) [CS12]. A move in these two papers positioned the sublist \( (\pi_1, …, \pi_w) \) immediately after the position of \( \pi_{i-1} \) where \( x \) is the element that immediately precedes \( \pi_{i-1} \) (if such an \( x \) exists) [CS12]. The symbols that are not \( \pi \) are denoted by \( \alpha, \beta \) which we call singles and doubles. Their sorting sequence for \( R_8 \) shown below creates seven new adjacencies (shown in bold) in six moves. Fortuna [F05] proved the correctness of this algorithm.

\[
\begin{align*}
(7,6,5,4) & \rightarrow (3,2) \rightarrow (1,7) \rightarrow (6,3) \rightarrow (2,5) \rightarrow (3,4) \rightarrow (0,1,2,3,4,5,6,7) \\
3,2,1,0 & \rightarrow 1,7,6,5,4,0 \rightarrow 6,3,2,5,4,0 \rightarrow 6,2,5,4,0,1 \rightarrow 2,5,6,3 \rightarrow 0,1,2,3,4,5,6,7
\end{align*}
\]

A similar sequence creates seven adjacencies in six moves for a permutation of the form \( (7, \alpha, 6, 5, 4, 3, 2, 1, 0, \beta) \), where \( \alpha \) and \( \beta \) are arbitrary sublists. In general, for any \( i \), one can employ this sequence on a permutation of the form \( (i, i+1, i+2, i+3, i+4, i+5, i+6, i+7, \alpha, \beta) \) which we call \( R'_{i, \alpha} \) to create seven adjacencies in six moves resulting in \( (i, i+1, i+2, i+3, i+4, i+5, i+6, i+7, \alpha, \beta) \) [2]. The last two symbols of \( \pi \) are denoted by \( x, y \) respectively. [CS08] gives an algorithm that executes greedy moves (singles) and culminates in \( (x+1, y-1, y, x, y, x, y, x, y) \). This move is a double because it creates two new adjacencies: \( (x+1, y) \) and \( (y, x, y) \). These singles are executed in such a way that the symbol in the first position of the resultant permutation is closest to \( x+1 \) [2]. Thus, \( (6)^{475*132} \rightarrow (475*132) \rightarrow (4756132) \) is a greedy move as it creates a new adjacency and makes 4 as the new \( \pi_1 \) which is closest to \( x+1 \) (i.e. \( x+1 \)).

In fact, the distance between \( x+1 \) and 4 is zero. Thus, the greedy method culminates in a double: \( (41)^{3*2} \rightarrow (3412) \).

Chitturi and Sudborough showed that at most \( k-1 < 7n/8 \) singles need be executed before one can execute a double [CS08]. \( \pi \) can be written as \( (t_1, …, s_i, t_2, …, s_j, …, t_l, …, s_k, t_{k+1}) \) where \( \forall j \not\in s_j \) is defined as the symbol immediately preceding \( t_j \) (if such an \( s_j \) exists) [CS12]. The symbols that are not \( \pi \) are called unvisited symbols. We call the sequence of symbols \( (t_1, t_1-1, t_1-2, t_1-3, …, x+1) \) or the set \( (t_1, t_1-1, t_1-2, t_1-3, …, x+1) \) as the forward chain [C15].
Its cardinality is a trivial upper bound for the number of visited symbols. An unvisited symbol \( r \) is called a bypassed symbol if \( r \) is forward chain; otherwise, it is a skipped symbol. We call \((t_1, \ldots, t_2)\) as the first interval, \((t_i, \ldots, t_{i+1})\) as the \( i \)th interval etc.. When \( \pi = s_1, \ldots, s_{l+1}, t_{k+1}, y \) transforms into \( \pi^* \) where \( \pi^* = (s_1, \ldots, s_{l+1}, t_{k+1}, l_{k+1}, l_{k+2}, \ldots, s_{k+1}, l_{k+1}, y) \), the symbols to the left of \( t_i \) in \( \pi \) are positioned in some intervals of \( \pi^* \). Such symbols that have been carried forward as the result of previous moves will not occur in the first position of a subsequent permutation; we call them as moved symbols. Further, we call the unvisited symbols among the moved symbols as omitted symbols. We call \( s_i \) as the leftmost if \( s_i \in (t_1, \ldots, t_2) \) and there is no skipped symbol in an interval \( j \) (\( 1 < j < i - 1 \)). Assuming that \( t_i = t_{i-1} - 1 \) for \( j \leq i \), the leftmost \( s_i \) (recall that \( s_i \) is the unvisited symbol immediately preceding \( t_i \)) has a maximum subscript of eight [CS08, CS12]; this is the justification for the value of \( k < 7n/8 \). Otherwise, \( R' \) will be a prefix.

2 Main Result

Observation 1: Chitturi [C15] claims in observation2 that for \( \alpha \) in \([0, 1]\), a sequence of moves that visits \( \alpha n \) symbols and skip \((1 - \alpha) n \) symbols when a double is executed, has an upper bound \( n - \log(\frac{1}{1-\alpha}) n \). If \( x \) is the number of visited symbols and \( y \) is the number of skipped symbols, then \( x = \alpha n \) and \( y = (1 - \alpha)n \).

\[
\frac{x}{y} = \frac{\alpha n}{(1-\alpha)n} \implies ay = (1-\alpha)x \implies a(x+y) = x \implies a = \frac{x}{x+y}. \text{ So, we have } \frac{1}{1-\alpha} = \frac{1}{\frac{x}{x+y}} = \frac{y+x}{y}
\]

Hence the base of the logarithm in the upper bound is \( \frac{\text{total number of moved symbols}}{\text{number of skipped symbols}} \).

\( \pi = t_1 t_2 \ldots t_{i-1} \ldots s_j t_i \ldots s_k t_k \ldots x y \) be the sequence discussed in Chitturi(2012), where \( t_i \)'s are the visited symbols and \( s_j \)'s are skipped symbols. An underscore is used on a symbol to denote that it is a skipped symbol in the sequence \( \pi \) with respect the sequence length algorithm. We execute greedy moves, i.e. create singles in each move until a double is realized. Furthermore, we consider the case when \( i \) is at most 8, else we shall execute \( R' \) [D02].

Observation 2: If more than 3 elements are skipped in at most 7 greedy moves, then we get an upper bound of \( n - \log(\frac{1}{1-\alpha}) n \) where both \( x, y \geq 0 \). Here more than 3 elements is denoted by \( 4+x \) and at most 7 greedy moves by \( 7-y \). RHS is maximized when the base of the logarithm is maximized this corresponds to \( y=0 \) yielding \( n - \log(\frac{1+x}{1+y}) n \). Likewise, \( x=0 \), yielding an upper bound of \( n - \log(\frac{1}{1+y}) n \). \( \square \)

In the sequel we make the following assumptions:
1. The first skipped symbol in \( \pi \) lies in the \( l \)th interval
2. \( s_j \) is the last skipped symbol in \( l \)th interval.
3. \( t_{l-1} = (s_j - l) \), for some \( l \geq 1 \) (if \( t_{l-1} = (s_j + l) \) for some \( l \geq 1 \), then \( s_j \) would be a visited symbol)
4. The \( l \)th interval has at most 3 skipped symbols. Otherwise, by Observation 2, \( n - \log(\frac{1}{1+y}) n \) is an upper bound.

Lemma 1: If \( i \leq 3 \) then the corresponding upper bound is \( n - \log(3) n \) prefix transpositions.

Proof: If \( i \leq 3 \), we skip at least one symbol in 2 regular greedy moves. By observation 1, the upper bound is \( n - \log(\frac{1}{1-\frac{1}{3}}) n = n - \log(3) n \). \( \square \)

Lemma 2: If \( i \geq 3 \) and \((s_j + 1)\) lies to the right of \( t_i \) then \( n - \log(\frac{1}{1+y}) n \) is an upper bound.

Proof: Consider the following move that moves at least \( i \) symbols of which \((i-1)\) are skipped
\[
[t_1 t_2 \ldots (s_i - l) \ldots s_i t_i \ldots * (s_i + 1) \rightarrow t_i \ldots t_1 t_2 \ldots (s_i - l) \ldots s_i (s_i + 1)]
\]

Hence by observation 1, the upper bound is \(n - \log_{\frac{4}{3}} n\) which maximises the base of the logarithm to \(\frac{4}{3}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\frac{5}{3}} n\). \(\square\)

**Lemma 3:** If \(i \geq 3\), \((s_i + 1)\) is a skipped symbol that lies to the left of \(t_i\) then \(n - \log_{(3)} n\) is an upper bound.

Proof: If \((s_i + 1)\) lies in the the \(j^{th}\) interval, where \(j < i\), then by assumption 3, we would have considered the symbol \(s_j\) rather than \(s_i\). So we shall assume that both \((s_i + 1)\) and \(s_i\) are skipped and lie in the \(i^{th}\) interval.

Let \(\pi = t_1 t_2 \ldots (s_i - l) \ldots (s_i + 1) \ldots s_i t_i \ldots s_k t_k \ldots x y\). Here we shall consider the position of \((s_i + 2)\).

**Case 1:** Suppose \((s_i + 2)\) lies to the right of \(t_i\), then in the following two moves

\[
[t_1 t_2 \ldots (s_i - l) \ldots (s_i + 1)] \quad s_i t_i \ldots * (s_i + 2) \quad \rightarrow \quad \cdots s_i t_i \ldots t_1 t_2 \ldots (s_i - l) \ldots (s_i + 1) \quad (s_i + 2)
\]

\[
[\ldots s_i \ldots t_i \ldots t_1 t_2 \ldots (s_i - l) \ldots * (s_i + 1) \quad (s_i + 2) \quad \rightarrow \quad t_i \ldots t_1 t_2 \ldots (s_i - l) \ldots s_i (s_i + 1) (s_i + 2)
\]

at least \((i + 1)\) symbols are moved of which \((i - 1)\) are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\frac{5}{i+1}} n\) which maximises the base of the logarithm to \(\frac{5}{3}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\frac{5}{3}} n\).

**Case 2:** Suppose \((s_i + 2)\) is visited and lies to the left of \(t_i\), then we consider the position of the symbol \((s_i - l + 1)\) in \(\pi\). Then by the sequence length algorithm by Chitturi(2012), \(\text{distance} (\{(s_i - l)\}, x) < \text{distance} (\{(s_i + 2)\}, x)\) and hence \((s_i + 2)\) lies to the left of \((s_i - l)\).

**Case 2.1** \((s_i - l + 1)\) lies to the right of \(t_i\). Here we need not consider the case where \(l = 1\), since then \((s_i - l + 1) = s_i\). Then in the following three moves

\[
[t_1 t_2 \ldots (s_i + 2) \ldots (s_i - l) \ldots s_i t_i \ldots * (s_i - l + 1)]
\]

\[
\rightarrow \quad \cdots (s_i + 1) \quad s_i t_i \ldots t_1 t_2 \ldots (s_i + 2) (s_i - l) (s_i - l + 1)
\]

\[
[\ldots (s_i + 1) \ldots s_i t_i \ldots t_1 t_2 \ldots * (s_i + 2) \ldots (s_i - l) (s_i - l + 1)]
\]

\[
\rightarrow \quad \cdots s_i t_i \ldots t_1 t_2 \ldots (s_i + 1) (s_i + 2) (s_i - l) (s_i - l + 1)
\]

at least \((i + 1)\) symbols are moved of which \((i - 2)\) are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\frac{5}{i+2}} n\) which maximises the base of the logarithm to \(\frac{5}{2}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\frac{5}{2}} n\).

**Case 2.2** \((s_i - l + 1)\) is skipped and lies to the left of \(t_i\).

4
(a) Suppose \( l \neq 1 \), if \((s_i - l + 1)\) is first symbol then we have an adjacency between \((s_i - l)\) and \((s_i - l + 1)\) and hence to avoid this there should be at least one more skipped symbol which makes the number of skipped symbols four. So, the permutation \( \pi = t_1 \ t_2 \ ... \ (s_i + 2) \ (s_i - l) \ (s_i + 1) \ (s_i - l + 1) \ s_i \ t_i \ ... \ x \ y \). Then in the following three moves

\[
\begin{align*}
&[t_1 \ t_2 \ ... (s_i + 2) \ (s_i - l)] \ (s_i + 1) \ (s_i - l + 1) \ s_i \ t_i \ ...

&\rightarrow (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ (s_i - l) \ (s_i - l + 1) \ s_i \ t_i \ ...

&\left( (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ (s_i - l) \ (s_i - l + 1) \ s_i * t_i \ ...

&\rightarrow (s_i - l) \ (s_i - l + 1) \ s_i \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ ... \ t_i \ ...

&\left( (s_i - l) \ (s_i - l + 1) \ s_i \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ ...

\right.

\]

at least \((i + 2)\) symbols are moved of which \((i - 1)\) are skipped. Hence by observation 1, the upper bound is given by \( n - \log_{ \frac{i+2}{i-2} } n \) which maximises the base of the logarithm to \( \frac{6}{3} \) when \( i = 4 \). Thus, the best upper bound is \( n - \log_{(3)} n \). Note that in this sequence of moves can be executed even if \((s_i - l - 1) = t_i\).

(b) If \( l = 1 \), Then in the following four moves.

\[
\begin{align*}
&[t_1 \ t_2 \ ... (s_i + 2) \ (s_i - 1)] \ c \ (s_i + 1) * s_i \ t_i \ ...

&\rightarrow c \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ (s_i - 1) \ s_i \ t_i \ ...

&\left( c \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ (s_i - 1) \ s_i \ t_i \ ...

&\rightarrow (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ (s_i - 1) \ s_i \ t_i \ ... \ (c + 1)

\right.

\]

(we make the above move irrespective of the position of \((c + 1)\), even though here \((c + 1)\) is assumed to lie to the right of \(t_i\)).

\[
\begin{align*}
&\left( (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ ...

&\rightarrow (s_i - 1) \ s_i \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ t_i \ ...

&\left( (s_i - 1) \ s_i \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ ...

&\rightarrow (s_i - 2) \ (s_i - 1) \ s_i \ (s_i + 1) \ t_1 \ t_2 \ ... (s_i + 2) \ ... \ t_i \ ...

at least \((i + 2)\) symbols are moved of which \((i - 2)\) are skipped. Hence by observation 1, the upper bound is given by \( n - \log_{ \frac{i+2}{i-2} } n \) which maximises the base of the logarithm to \( \frac{6}{2} \) when \( i = 4 \). Thus, the best upper bound is \( n - \log_{(3)} n \). Note that if \( c \) does not exist or lie between \((s_i + 1)\) and \( s_i \) then the second move can be omitted, and hence we obtain a better upper bound. Further this sequence of moves can be executed even if \((s_i - 2) = t_i\).

Case 2.3 \((s_i - l + 1)\) is visited and lies to the left of \(t_i\). Here \( \pi = t_1 \ t_2 \ ... (s_i + 2) \ ... (s_i - l + 1) \ (s_i - l) \ (s_i + 1) * t_i \ ... \ s_i \ t_i \ ... \ x \ y \). Further \((s_i - l + 1)\) will lie immediately to the left of \((s_i - l)\), otherwise if there is another visited symbol between them, then \((s_i - l)\) would be skipped by the construction of sequence length algorithm. Then we follow the usual greedy moves for all visited symbols until \((s_i + 2)\) becomes the first symbol. Then we do the following moves.
\[
\left[ (s_i + 2) \ldots (s_i - l + 1) \right] (s_i - l) \ldots (s_i + 1) \ast \ldots s_i t_i \\
\rightarrow (s_i - l) \ldots (s_i + 1) (s_i + 2) \ldots (s_i - l + 1) s_i t_i \\
\left[ (s_i - l) \ldots (s_i + 1) (s_i + 2) \ldots (s_i - l + 1) s_i \right] t_i \ldots (s_i - l - 1) \ldots \ldots \ldots (s_i + 1) (s_i + 2) \ldots (s_i - l + 1) s_i \\
\rightarrow t_i \ldots (s_i - l - 1) (s_i - l) \ldots (s_i + 1) (s_i + 2) \ldots (s_i - l + 1) s_i \\
\]

Here in at most \((i - 2)\) moves at least \((i + 1)\) symbols are moved of which three are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\left(\frac{i+1}{3}\right)} n\) which maximises the base of the logarithm to \(\frac{9}{3}\) when \(i = 8\). Thus, the best upper bound is \(n - \log_{(2)} n\).

**Case 3**: Suppose \((s_i + 2)\) is skipped and lies to the left of \(t_i\), then we consider the position of the symbol \((s_i + 3)\) in \(\pi = t_1 t_2 \ldots (s_i - l) (s_i + 2) (s_i + 1) s_i t_i \ldots x y\).

**Case 3.1** \((s_i + 3)\) lies to the right of \(t_i\). Then in the following three moves

\[
\left[ t_1 t_2 \ldots (s_i - l) (s_i + 2) \right] (s_i + 1) s_i t_i \ldots (s_i + 3) \rightarrow (s_i + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i - l) (s_i + 2) (s_i + 3) \\
\left[ (s_i + 1) \right] s_i t_i \ldots t_1 t_2 \ldots (s_i - l) \ldots (s_i + 2) \ldots (s_i + 3) \rightarrow s_i t_i \ldots t_1 t_2 \ldots (s_i - l) (s_i + 1) (s_i + 2) (s_i + 3) \\
\left[ s_i \right] t_i \ldots t_1 t_2 \ldots (s_i - l) \ldots (s_i + 1) (s_i + 2) (s_i + 3) \ldots \rightarrow t_i \ldots t_1 t_2 \ldots (s_i - l) s_i (s_i + 1) (s_i + 2) (s_i + 3) \\
\]

at least \((i + 2)\) symbols are moved of which \((i - 1)\) are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\left(\frac{i+2}{2}\right)} n\) which maximises the base of the logarithm to \(\frac{6}{3}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{(2)} n\).

**Case 3.2** \((s_i + 3)\) is visited and lies to the left of \(t_i\). Then we consider the position of the symbol \((s_i - l + 1)\) in \(\pi\).

(a) \((s_i - l + 1)\) lies to the right of \(t_i\). Here we need not consider the case where \(l = 1\), since then \((s_i - l + 1) = s_i\). Then in the following four moves

\[
\left[ t_1 t_2 \ldots (s_i + 3) \ldots (s_i - l) \right] (s_i + 2) (s_i + 1) s_i t_i \ldots (s_i - l + 1) \ldots \\
\rightarrow (s_i + 2) (s_i + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i + 3) \ldots (s_i - l) (s_i - l + 1) \ldots \\
\left[ (s_i + 2) \right] (s_i + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i + 3) \ldots (s_i - l + 1) \ldots \\
\rightarrow (s_i + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i + 2) (s_i + 3) \ldots (s_i - l) (s_i - l + 1) \ldots \\
\left[ (s_i + 1) \right] s_i t_i \ldots t_1 t_2 \ldots (s_i + 3) \ldots (s_i - l) (s_i - l + 1) \ldots \\
\rightarrow s_i t_i \ldots t_1 t_2 \ldots (s_i + 1) (s_i + 2) (s_i + 3) \ldots (s_i - l) (s_i - l + 1) \ldots \\
\left[ s_i \right] t_i \ldots t_1 t_2 \ldots (s_i + 1) (s_i + 2) (s_i + 3) \ldots (s_i - l) (s_i - l + 1) \ldots \\
\rightarrow t_i \ldots t_1 t_2 \ldots s_i (s_i + 1) (s_i + 2) (s_i + 3) \ldots (s_i - l) (s_i - l + 1) \ldots \\
\]

6
at least \((i + 2)\) symbols are moved of which \((i - 2)\) are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\left(\frac{i+2}{i+3}\right)} n\) which maximises the base of the logarithm to \(\frac{5}{2}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\left(\frac{5}{3}\right)} n\).

(b) \((s_l - l + 1)\) is visited and lies to the left of \(t_i\). Then we follow the usual greedy moves for all visited symbols until \((s_l + 3)\) becomes the first symbol. Then we do the following move

\[
\left[ (s_l + 3)(s_l - l + 1)(s_l - l)(s_l + 2)(s_l + 1) s_l t_i \ldots \rightarrow (s_l - l)(s_l + 2)(s_l + 3)(s_l - l + 1)(s_l + 1) s_l t_i \ldots \right.
\]

\[
\left. (s_l - l)(s_l + 2)(s_l + 3)(s_l - l + 1)(s_l + 1)s_l t_i \ldots \rightarrow t_i \ldots (s_l - l - 1)(s_l - l)(s_l + 2)(s_l + 3)(s_l - l + 1)(s_l + 1)s_l \ldots \right.
\]

Here in at most \((i - 2)\) moves at least \((i + 2)\) symbols are moved of which four are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\left(\frac{5}{i+2}\right)} n\) which maximises the base of the logarithm to \(\frac{5}{2}\) when \(i = 8\). Thus, the best upper bound is \(n - \log_{\left(\frac{5}{3}\right)} n\).

(c) \((s_l - l + 1)\) is skipped and lies to the left of \(t_i\). Here we explore only the case when \(l = 1\) other wise if \(l \neq 1\), the number of skipped symbols exceed three. Then by Observation 2 the upper bound is \(n - \log_{\left(\frac{5}{4}\right)} n\).

Now \(\pi = t_1 t_2 \ldots (s_l + 3)(s_l - 1)(s_l + 2)(s_l + 1)s_l t_i \ldots s_k t_k \ldots x y\). Consider two subcases here:

(i) When \((s_l + 4)\) lies to the right of \(t_i\), in the following two moves, \((i + 2)\) symbols are moved of which \(i\) are skipped.

\[
\left[ t_1 t_2 \ldots (s_l + 3)(s_l - 1)(s_l + 2)(s_l + 1)s_l t_i \ldots * (s_l + 4) \rightarrow (s_l - 1)(s_l + 2)(s_l + 1)s_l t_i \ldots t_i t_2 \ldots (s_l + 3)(s_l + 4) \right.
\]

\[
\left. (s_l - 1)(s_l + 2)(s_l + 1)s_l t_i \ldots t_i t_2 \ldots (s_l + 3)(s_l + 4)(s_l - 2)(s_l - 1)(s_l + 2)(s_l + 1)s_l \ldots \right.
\]

Hence by observation 1, the upper bound is given by \(n - \log_{\left(\frac{5}{i+2}\right)} n\) which maximises the base of the logarithm to \(\frac{5}{4}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\left(\frac{5}{3}\right)} n\). Further note that this sequence of moves can be executed even if \((s_l - 2) = t_i\).

(ii) When \((s_l + 4)\) is visited and lies to the left of \(t_i\). Then \(\pi = t_1 t_2 \ldots (s_l + 4)(s_l + 3)(s_l - 1)(s_l + 2)(s_l + 1)s_l t_i \ldots x y\). Then in the following four moves

\[
\left[ t_1 t_2 \ldots (s_l + 4)(s_l + 3)(s_l - 1)(s_l + 2)(s_l + 1)s_l t_i \ldots \rightarrow (s_l + 2)(s_l + 1)t_2 \ldots (s_l + 4)(s_l + 3)(s_l - 1)s_l t_i \ldots \right.
\]

\[
\left. (s_l + 1)t_2 \ldots (s_l + 4)(s_l + 3)(s_l - 1)s_l t_i \ldots \rightarrow (s_l + 1)(s_l + 2)t_2 \ldots (s_l + 4)(s_l + 3)(s_l - 1)s_l t_i \ldots \right.
\]

\[
\left. (s_l + 1)(s_l + 2)t_2 \ldots (s_l + 4)(s_l + 3)(s_l - 1)s_l t_i \ldots \rightarrow (s_l - 1)s_l (s_l + 1)(s_l + 2)t_2 \ldots (s_l + 4)(s_l + 3)t_l \ldots \right.
\]
\[
\left(\frac{s_i}{2} - 1\right) s_i \left(\frac{s_i}{2} + 1\right) \left(\frac{s_i}{2} + 2\right) t_1 t_2 \ldots \left(\frac{s_i}{2} + 4\right) \left(\frac{s_i}{2} + 3\right) t_i \ldots \left(\frac{s_i}{2} - 2\right) \ldots
\]

\[
\rightarrow t_i \ldots \left(\frac{s_i}{2} - 2\right) \left(\frac{s_i}{2} - 1\right) s_i \left(\frac{s_i}{2} + 1\right) \left(\frac{s_i}{2} + 2\right) t_1 t_2 \ldots \left(\frac{s_i}{2} + 4\right) \left(\frac{s_i}{2} + 3\right) \ldots
\]

at least \((i + 2)\) symbols are moved of which \((i - 2)\) are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{\left(\frac{1}{2}\right)} n\) which maximises the base of the logarithm to \(\frac{6}{2}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\left(\frac{3}{2}\right)} n\).

Hence from case 2.3 in the Lemma the upper bound for \(i \geq 3\) when \(s_i + 1\) is a skipped symbol and lies to the left of \(t_i\) is \(n - \log_{\left(\frac{3}{2}\right)} n\). □

**Lemma 4:** If \(i \geq 3\), \((s_i + 1)\) is a visited symbol and lies to the left of \(t_i\) and then \(n - \log_{\left(\frac{3}{2}\right)} n\) is an upper bound.

**Proof:** \(\pi = t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) \ldots s_l t_1 \ldots s_k t_k \ldots x y\). Here we shall consider the position of the symbol \((s_i - l + 1)\) in \(\pi\).

**Case 1:** Suppose \((s_i - l + 1)\) lies to the right of \(t_i\). Here we need not consider the case where \(l = 1\), since then \((s_i - l + 1) = s_i\). Then in the following two moves

\[
\left[t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) \ldots s_l t_1 \ldots * (s_i - l + 1) \rightarrow \ldots s_l t_1 \ldots t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) \ldots (s_i - l + 1) \ldots
\]

\[
\left[\ldots s_l t_1 \ldots t_1 t_2 \ldots * (s_i + 1) \ldots (s_i - l) \ldots (s_i - l + 1) \ldots \rightarrow t_l \ldots t_1 t_2 \ldots s_l (s_i + 1) \ldots (s_i - l) \ldots (s_i - l + 1) \ldots
\]

at least \(i\) symbols are moved of which \((i - 2)\) are skipped. Hence by observation 1, the upper bound is \(n - \log_{\left(\frac{1}{2}\right)} n\) which maximises the base of the logarithm to \(\frac{4}{2}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\left(\frac{3}{2}\right)} n\).

**Case 2:** Suppose \((s_i - l + 1)\), is visited and lies to the left of \(t_i\). Here we need not consider the case where \(l = 1\), since then \((s_i - l + 1) = s_i\). Consider the symbol \((s_i - l + 2)\) in \(\pi\).

**Case 2.1** \((s_i - l + 2)\) lies to the right of \((s_i - l + 1)\). Here \((s_i - l + 2)\) may be a skipped symbol in the \(i^{th}\) interval or lie to the right of \(t_i\). Then in the following two moves

\[
\left[t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l + 1) \ldots (s_i - l) \ldots * (s_i - l + 2) \ldots
\]

\[
\rightarrow (s_i - l) \ldots t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l + 1) \ldots (s_i - l + 2) \ldots
\]

\[
\left[(s_i - l) \ldots s_l \ldots t_1 \ldots (s_i - l - 1) \ldots * \rightarrow t_l \ldots (s_i - l - 1) (s_i - l) \ldots
\]

at least \(i\) symbols are moved of which \((i - 2)\) are skipped. Hence by observation 1, the upper bound is \(n - \log_{\left(\frac{1}{2}\right)} n\) which maximises the base of the logarithm to \(\frac{4}{2}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{\left(\frac{3}{2}\right)} n\).

Further note that this sequence of moves can be executed even if \((s_i - l - 1) = t_i\).

**Case 2.2** \((s_i - l + 2)\) is visited and lies to the left of \((s_i - l + 1)\). Follow the usual greedy moves for all visited symbols until \((s_i + 1)\) becomes the first symbol. Then we do the following move
\[
\left[ (s_i + 1) \ldots (s_i - l + 2) \right] (s_i - l + 1) (s_i - l) \ldots s_i * t_i \ldots \\
\rightarrow (s_i - l) \ldots s_i (s_i - l + 1) \ldots (s_i - l + 2) \left( s_i - l + 1 \right) t_i \ldots
\]

where at least 2 visited symbols are skipped. So here in at most \((i - 3)\) moves at least \(i\) symbols are moved of which three are skipped. Hence by observation 1, the upper bound is \(n - \log\left(\frac{1}{3}\right) n\) which maximises the base of the logarithm to \(\frac{3}{8}\) when \(i = 8\). Thus, the best upper bound is \(n - \log\left(\frac{1}{3}\right) n\).

**Case 3:** Suppose \((s_i - l + 1)\) is skipped and lies to the left of \(t_i\). Then \((s_i - l + 1)\) cannot be the first skipped element, for then \((s_i - l) (s_i - l + 1)\) would be consecutive symbols in the permutation and hence form an adjacency. In this case we shall find an upper bound for two different cases-(1) when \((s_i - l + 1) \neq s_i\) (when \(l \neq 1\)) and (2) \((s_i - l + 1) = s_i\)

Then \(\pi = t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) \leq (s_i - l + 1) s_i t_i \ldots s_k t_k \ldots x y \). Consider the symbol \((s_i - l + 2)\) in \(\pi\).

**Case 3.1:** \((s_i - l + 2)\) lies to the right of \(t_i\). Consider the following two moves

\[
\left[ (s_i - l) \leq (s_i - l + 1) \right] (s_i - l) \leq (s_i - l + 1) s_i t_i \ldots \\
\rightarrow s_i t_i \ldots t_2 \ldots (s_i + 1) \ldots (s_i - l) \leq (s_i - l + 1) (s_i - l + 2)
\]

\[
\left[ s_i \right] t_i \ldots t_2 \ldots (s_i + 1) \ldots (s_i - l) \leq (s_i - l + 1) (s_i - l + 2) \\
\rightarrow t_i \ldots t_3 t_2 \ldots s_i (s_i + 1) \ldots (s_i - l) \leq (s_i - l + 1) (s_i - l + 2)
\]

\((i + 2)\) symbols are moved of which \(i\) are skipped. Hence by observation 1, the upper bound is \(n - \log\left(\frac{1}{3}\right) n\) which maximises the base of the logarithm to \(\frac{6}{4}\) when \(i = 4\). Thus, the best upper bound is \(n - \log\left(\frac{1}{3}\right) n\).

**Case 3.2:** \((s_i - l + 2)\) is visited and lies to the left of \(t_i\). Follow the usual greedy moves for all visited symbols until \((s_i + 1)\) becomes the first symbol. Then we do the following move where at least one visited symbol is skipped.

\[
\left[ (s_i + 1) \ldots (s_i - l + 2) \right] (s_i - l) \leq (s_i - l + 1) s_i t_i \ldots \\
\rightarrow (s_i - l) \leq (s_i - l + 1) s_i (s_i + 1) \ldots (s_i - l + 2) t_i \ldots
\]

\[
\left[ (s_i - l) \leq (s_i - l + 1) s_i (s_i + 1) \ldots (s_i - l + 2) \right] t_i \ldots (s_i - l - 1) * \ldots \\
\rightarrow t_i \ldots (s_i - l - 1) (s_i - l) \leq (s_i - l + 1) s_i (s_i + 1) \ldots (s_i - l + 2) \ldots
\]

Note that we can execute these moves even when \((s_i - l - 1) = t_i\). So here in at most \((i - 2)\) moves at least \((i + 2)\) symbols are moved of which four are skipped. Hence by observation 1, the upper bound is \(n - \log\left(\frac{1}{3}\right) n\) which maximises the base of the logarithm to \(\frac{10}{4}\) when \(i = 8\). Thus, the best upper bound is \(n - \log\left(\frac{1}{3}\right) n\).

**Case 3.3:** \((s_i - l + 2)\) is skipped and lies to the left of \(t_i\).
Then \( \pi = t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) (s_i - l + 2) (s_i - l + 1) s_i t_i \ldots s_k t_k \ldots x y \). The symbol \((s_i - l + 3)\) will lie either to the right of \(t_i\) or shall be a visited symbol after \((s_i + 1)\). Note: if \((s_i - l + 3) = (s_i + 1)\) then \((s_i - l + 2) = s_i\) and hence \((s_i - l + 1) (s_i - l + 2)\) would form an adjacency.

(a) \((s_i - l + 3)\) will lie to right of \(t_i\). Then in the following three moves
\[
[t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) (s_i - l + 2) (s_i - l + 1) s_i t_i \ldots (s_i - l) (s_i - l + 2) (s_i - l + 3) \\
\rightarrow (s_i - l + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) (s_i - l + 2) (s_i - l + 3) \\
\rightarrow (s_i - l + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) (s_i - l + 1) (s_i - l + 2) (s_i - l + 3)
\]

\((i + 2)\) symbols are moved of which \((i - 1)\) are skipped. Hence by observation 1, the upper bound is given by \(n - \log_{(i+2)} n\) which maximises the base of the logarithm to \(\frac{6}{3}\) when \(i = 4\). Thus, the best upper bound is \(n - \log_{(i+2)} n\).

(b) If \((s_i - l + 3)\) is a visited symbol after \((s_i + 1)\). Then \(\pi = t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l + 3) (s_i - l) (s_i - l + 2) (s_i - l + 1) s_i t_i \ldots s_k t_k \ldots x y \). Here we follow the usual greedy moves for all visited symbols until \((s_i + 1)\) becomes the first symbol. Then we do the following move where at least one visited symbol is skipped.
\[
[(s_i + 1) \ldots (s_i - l + 3) (s_i - l) (s_i - l + 2) (s_i - l + 1) s_i t_i \ldots (s_i - l) (s_i - l + 2) (s_i - l + 3) \\
\rightarrow (s_i - l + 1) s_i t_i \ldots t_1 t_2 \ldots (s_i + 1) \ldots (s_i - l) (s_i - l + 1) (s_i - l + 2) (s_i - l + 3)
\]

Note that we can execute these moves even when \((s_i - l - 1) = t_i\). So here in at most \((i - 2)\) moves at least \((i + 2)\) symbols are moved of which four are skipped. Hence by observation 1, the upper bound is \(n - \log_{(i+2)} n\) which maximises the base of the logarithm to \(\frac{10}{4}\) when \(i = 8\). Thus, the best upper bound is \(n - \log_{(i+2)} n\).

Now we shall consider the second part of the proof where \((s_i - l + 1) = s_i\).

Clearly \(s_i\) is not the only skipped symbol in the \(i^{th}\) interval because then there would be an adjacency between \((s_i - 1)\) and \(s_i\). Let \(c\) be the first skipped symbol in the \(i^{th}\) interval. Then \(\pi = t_1 t_2 \ldots (s_i + 1) \ldots (s_i - 1) c \ldots s_i t_i \ldots s_k t_k \ldots x y \).

(a) If \((c + 1)\) lies to the right of \(t_i\). Then in the following two moves
\[
[t_1 t_2 \ldots (s_i + 1) \ldots (s_i - 1) c \ldots s_i t_i \ldots (c + 1) \rightarrow \ldots s_i t_i \ldots t_1 t_2 \ldots (s_i + 1) \ldots (s_i - 1) c (c + 1)\]

... $s_i \ t_1 \ t_2 \ ... * (s_i + 1) \ ... \ (s_i - 1) \ c \ (c + 1) \ ... \ t_i \ t_1 \ t_2 \ ... s_i (s_i + 1) \ ... \ (s_i - 1) \ c \ (c + 1) \ ...$

at least $(i + 1)$ symbols are moved of which $(i - 1)$ are skipped. Hence by observation 1, the upper bound is given by $n - \log \left(\frac{3}{i-1}\right)n$ which maximises the base of the logarithm to $\frac{5}{3}$ when $i = 4$. Thus, the best upper bound is $n - \log \left(\frac{3}{5}\right)n$.

(b) If $(c - 1)$ lies to the right of $t_i$. Then in the following two moves

$[t_1 \ t_2 \ ... \ (s_i + 1) \ ... \ (s_i - 1) \ c \ ... \ s_i t_i \ ... \ (c - 1) \ c \ ... \ t_1 \ t_2 \ ... \ (s_i + 1) \ ... \ (s_i - 1) \ s_i \ c \ ...]$

at least $(i + 1)$ symbols are moved of which $(i - 1)$ are skipped. Hence by observation 1, the upper bound is given by $n - \log \left(\frac{3}{5}\right)n$ which maximises the base of the logarithm to $\frac{5}{3}$ when $i = 4$. Thus, the best upper bound is $n - \log \left(\frac{3}{5}\right)n$.

(c) If $(c + 1)$ and $(c - 1)$ lies to the left of $t_i$, if $(c + 1)$ is a skipped element, then there should be at least one more skipped symbol between $c$ and $(c + 1)$, else we get an adjacency $c (c + 1)$ in $\pi$. So, then the number of skipped elements becomes four. Hence, we shall assume that $(c + 1)$ is a visited symbol. Now we shall consider two subcases according to the position of $(c - 1)$ in $\pi$.

(i) $(c - 1)$ is visited. Then $\pi = t_1 \ t_2 \ ... \ (c + 1)(c - 1) \ c \ ... \ s_i \ t_i \ ... \ s_k \ t_k \ ... \ x \ y$. The first move that we execute here is

$[t_1 \ t_2 \ ... \ (c + 1)(c - 1) \ t_j \ ... \ (s_i - 1) \ c \ ... \ s_i t_i \ ... \ t_j \ ... \ (s_i - 1) \ t_1 \ t_2 \ ... \ (c + 1)(c - 1) \ c \ ... \ s_i t_i \ ...]$

This move skips at least one visited symbol (namely $(c + 1)$). Later we execute the regular greedy moves from the visited symbol $t_j$. In this sequence of moves, at most $(i - 2)$ moves we move at least $(i + 1)$ symbols and skip three. Hence by observation 1, the upper bound is given by $n - \log \left(\frac{3}{i-1}\right)n$ which maximises the base of the logarithm to $\frac{5}{3}$ when $i = 8$. Thus, the best upper bound is $n - \log \left(\frac{3}{5}\right)n$.

(ii) $(c - 1)$ is skipped. Then $\pi = t_1 \ t_2 \ ... \ (c + 1) \ ... \ (s_i + 1) \ ... \ (s_i - 1) \ c (c - 1) \ s_i t_i \ ... \ s_k \ t_k \ ... \ x \ y$. Here we follow the regular greedy moves until $(c + 1)$ becomes the first symbol. The next move is

$[(c + 1) \ ... \ (s_i + 1)] \ t_j \ ... \ (s_i - 1) \ c \ ... \ (c - 1) s_i t_i \ ... \ t_j \ ... \ (s_i - 1) \ c (c + 1) \ ... \ (s_i + 1) \ (c - 1) s_i t_i \ ...$

which skips at least one visited symbol (namely $(s_i + 1)$). Later we execute the regular greedy moves from the visited symbol $t_j$. By this sequence of moves, in at most $(i - 2)$ moves we move $(i + 2)$ symbols and skip four. Hence by observation 1, the upper bound is given by $n - \log \left(\frac{3}{i+2}\right)n$ which maximises the base of the logarithm to $\frac{10}{4}$ when $i = 8$. Thus, the best upper bound is $n - \log \left(\frac{3}{5}\right)n$.

By considering all the cases discussed in Lemma 4, the upper bound for $i \geq 3$ where $(s_i + 1)$ is a visited symbol and lies to the left of $t_i$ is $n - \log g(3)n$.

**Theorem 1:** $n - \log (3)n$ is an upper bound to sort permutations with prefix transpositions.

**Proof:** If $R_i'$ is a prefix of $\pi$ then the corresponding upper bound is $3n/4$. If $R_i'$ is not a prefix, we encounter at least one skipped symbol $s_i$, $1 \leq i \leq 8$. If more than 3 elements are skipped in at most 7 greedy moves, then by
Observation 2 we obtain an upper bound of $n - log(\frac{1}{4})n$. If the number of skipped symbols is at most three, Lemmas 1, 2, 3 and 4 prove an upper bound of $n - log(\frac{1}{3})n$. Hence the theorem.

3 Conclusion
The existing upper bound of $n - log_{\frac{7}{2}}n$ to sort any $\pi \in S_n$ with prefix transpositions has been improved to $n - log_3n$.

References