Exact Upper Bound for Sorting R_n with LE

Sai Satwik Kuppili[1] and Bhadrachalam Chitturi[1,2]

[1] Dept. of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India
[2] Dept. of Computer Science, University of Texas at Dallas, Texas, USA
satwik.kuppili@gmail.com, chalam@utdallas.edu

Abstract. A permutation over alphabet $\Sigma = (1, 2, 3, \ldots, n)$ is a sequence over Σ where every element occurs exactly once. S_n denotes symmetric group defined over Σ. $I_n = (1, 2, 3, \ldots, n) \in S_n$ denotes the Identity permutation. $R_n \in S_n$ is the reverse permutation i.e. $R_n = (n, n-1, n-2, \ldots, 2, 1)$. An operation has been defined in OEIS which consists of exactly two moves: set-rotate that we call Rotate and pair-exchange that we call Exchange. Rotate is a left rotate of all elements (moves leftmost element to the right end) and Exchange is the pair-wise exchange of the two leftmost elements. We call this operation as LE. The optimum number of moves for transforming R_n into I_n with LE operation are known for $n \leq 10$; as listed in OEIS with identity A048200. The contributions of this article are: (a) a novel upper bound for the number of moves required to sort R_n with LE has been derived; (b) the optimum number of moves to sort the next larger R_n i.e. R_{11} has been computed; (c) an algorithm conjectured to compute the optimum number of moves to sort a given R_n has been designed.

Keywords: Permutations · Sorting · Cayley graphs · Upper bound · Set-Rotate · Pair-Exchange.

1 Introduction

[3] Sorting a permutation can be either in an increasing or a decreasing order. In this article, increasing order is employed. The alphabet for the permutations is $\Sigma = (1, 2, 3, \ldots, n)$. LE operation consists of two generators: (i) *Rotate* that cyclically left shifts the entire permutation and (ii) *Exchange* that swaps the elements at the two left most positions. The application of (i) and (ii) yields the corresponding *moves* L and E respectively. 1-based indexing is employed, thus, the two leftmost indices are one and two. S_n is the set of all permutations over Σ. R_n is the reverse permutation of Σ, i.e. $R_n = (n, n-1, \ldots, 3, 2, 1)$. The identity permutation $I_n = (1, 2, 3, \ldots, n-1, n)$. The problem of transforming R_n into I_n (i.e. sorting R_n) with LE operation is of theoretical interest and has been studied. The problem appears in OEIS [2] as follows "A048200 Minimal length pair-exchange/set-rotate sequence to reverse n distinct ordered elements".

[3] A preliminary version of this article appears in ICACDS 2019 [1].

The optimum number of moves to sort R_n with LE appears with a sequence number of A048200, OEIS [2] where the values are known only for $n \leq 11$ [2] ($n = 11$ is our contribution). We establish the first upper bound on the number of moves required to sort R_n with LE.

A Cayley graph Γ corresponding to an operation O with a generator set G consists of $n!$ vertices each corresponding to a unique permutation that denotes it. An edge from a vertex u to another vertex v indicates that when a generator $g \in G$ acts upon permutation u yields v. Applying a generator is commonly known as making a *move*. An upper bound k to sort any $\pi \in S_n$ indicates that the distance between any two permutations in Γ is at most k. An exact upper bound equals the diameter of Γ [4]. Cayley graphs are shown to posses various desirable properties in the design of computer interconnection networks [4, 9]. Various operations to sort permutations have been posed [9]. A permutation models a genome where a gene is presumed to be unique and an operation like transposition, reversal etc. models the corresponding mutation. Thus, transforming permutations with various operations has applications in genomic studies.

Jerrum showed that when the number of generators is greater than one, the minimum sequence of generators (also called as distance) to sort a permutation is hard to compute [3]. LE operation has two generators and the complexity of transforming one permutation in to another with LE operation is not known. Exchange move is a reversal of length two, in fact it is a prefix reversal of length two. Chen and Skiena studied sorting permutations of length n with reversals of size p [6]. For both permutations and circular permutations for all n and p, they characterized the number of equivalence classes of permutations. For sorting all circular permutations of length n that can be sorted by reversals of length p an upper bound of $O(n^2/p + pn)$ and a lower bound of $\Omega(n^2/p^2 + n)$ were shown. For sorting permutations with (unrestricted) prefix reversals the operation that has $n-1$ generators, the best known upper bound is $18n/11 + O(1)$ [5]. In LE operation, Rotate cyclically shifts the entire permutation whereas in [7] a modified bubblesort is considered, where, in addition to the regular moves, a swap is allowed between elements at positions 1 and n. Given an operation O, all the moves of O constitute its generator set. Jerrum showed that when the number of generators is greater than one, the minimum sequence of generators to sort a permutation is hard to compute [3]. LE operation has two generators and the complexity of transforming one permutation into another with LE operation is not known. We call O symmetric if for any move of O its *inverse* also belongs to O. Exchange is inverse of itself whereas Rotate does not have an inverse. Thus, LE is not symmetric. Further, LE is very restrictive due to the presence of Rotate move compared to the other operations that are frequently applied in genetic studies e.g. [8]. The methodology of this article might be helpful for problems whose generator set does not have a Rotate generator. Research in the area of Cayley graphs pertaining to their efficacy in modelling a computer interconnection network, their properties in terms of diameter, presence of greedy cycles in them etc. has been active [12–16].

Two permutations are *equivalent* if one can be transformed into another by applying a finite number of Rotate moves. In order to show that LE operation generates the entire symmetric group S_n, we need only show that any two elements can be swapped.

Transformation of strings also has been extensively studied [17, 18]. Several string transformation problems including the burnt pancake distance problem are shown to be NP-hard [18]. An operation called as short reversal on strings has been defined [8] that has exactly two types of generators. The computation of short reversal distance has been reduced to the computation of a Maximum Independent Set on the corresponding graph that is computed from the two given input strings [10] an efficient algorithm for it has been designed in [11].

Observation 1 *Any two elements can be swapped with LE operation.*

Proof. Consider two arbitrary elements a and b in a permutation π. WLOG assume that a is to the left of b. So $\pi = (\ldots, u, a, v, \ldots, x, b, y, \ldots)$. First we perform a sequence of Rotate moves to yield $(a, v, \ldots, x, b, y, \ldots, u)$. Here we perform a sequence of (Exchange followed by Rotate) to yield $(a, b, y, \ldots, u, v, \ldots, x,)$. After Rotate, Exchange, Rotate this yields $(b, \ldots, u, v, \ldots, x, a, y)$ where a is between x, y. We follow the same procedure to place b between u and v. Then we will get a permutation that is equivalent to the permutation in which a and b are swapped. Rotate moves accomplish the rest of the task.

Let π be the one based index array containing the input permutation. The element at an index i of π is denoted by $\pi[i]$. Initially for all i, $\pi[i] = R_n[i]$. A *block* is a sublist (continuous elements of a permutation) that is sorted. Let EL denote Exchange move followed by a Rotate move. Further, let $(EL)^p$ be p consecutive executions of EL. Let L^p be p consecutive executions of L. We define a permutation $P_{r,n} \in S_n$ as follows. The elements $1, 2, 3, \ldots, r$ are in sorted order. However, $(n, n-1, \ldots, r+1)$ that we call $U(P_{r,n})$ is inserted in between. Thus, $(1, 2, 3, \ldots, r)$ is split into two blocks with $U(P_{r,n})$ in between. Further, the starting position of $U(P_{r,n})$ in $P_{r,n}$ is $x + 2$ where $r = 2^k - x$ and $2^{k-1} < r \leq 2^k$. Let $M(n)$ be the number of moves required to sort R_n with LE. Let $f(x)$ denote the number of additional moves required to sort R_n with LE when compared to R_{n-1}. Therefore, number of moves required to sort R_n with LE is sum of all $f(x)$ where x ranges from 1 to n.

2 Algorithm

The algorithm that we design is called Algorithm LE. The algorithm first transforms R_n into $P_{3,n}$ by executing $n-2$ L moves and an E move. Subsequently, $P_{i+1,n}$ is obtained from $P_{i,n}$ by executing the moves specified by Lemma 2. Thus, eventually we obtain $P_{n,n}$ which is I_n. Pseudo Code for the Algorithm LE is shown below.

Algorithm LE

Input: R_n. Output: I_n. Initialization:$\forall i$ $\pi[i] = R_n[i]$. All moves are executed on π.

Algorithm 1 Algorithm LE

for $r \in (2, \ldots, n-1)$ **do**
  if r=2 **then**
    Execute L^{n-2}
    Execute E move
  else
    if $r = 2^k$ for some k **then**
      Execute $(EL)^{n-r}$
    else
      $x \leftarrow (\min_k \text{ s.t. } 2^k \geq r)$
      Execute L^{x-r}
      Execute $(EL)^{n-r-1}$
      Execute L move
      Execute $(EL)^{2r-x-1}$
      Execute L move
    end if
  end if
end for

Lemma 1. *The starting position of $U(P_{r,n})$ in $P_{r,n}$ is $x+2$ where $r = 2^k - x$ and $2^{k-1} < r \leq 2^k$.*

Proof. Executing $n-2$ L moves and an E moves on R_n yields $(1, 2, n, n-1, \ldots, 3)$ which is $P_{3,n}$. Since, $3 = 2^2 - 1$, $x = 1$. Thus, $x + 2 = 1 + 2 = 3$. If we observe the starting position of $U(P_{3,n})$ in $P_{3,n}$ is $x + 2 = 3$. Hence, lemma is true for $r = 3$. Assume, that lemma is true for $r = 2^k - x$. So, $P_{r,n}$ is $(1, 2, \ldots, x+1, n, n-1, \ldots, r+1, x+2, x+3, \ldots, r)$ where starting position of $U(P_{r,n})$ in $P_{r,n}$ is $x+2$ where $r = 2^k - x$ and $2^{k-1} < r \leq 2^k$. Executing L^x yields $(x+1, n, n-1 \ldots, r+1, x+2, x+3, \ldots, r, 1, 2, \ldots, x)$. Then executing $(EL)^{n-r-1}$ yields $(x+1, r+1, x+2, x+3, \ldots, r, 1, 2, \ldots, x, n, n-1, \ldots, r+2)$. Then executing R yields $(r+1, x+2, \ldots, r, 1, 2, \ldots, x, n, n-1, \ldots, r+2, x+1)$. Then executing $(EL)^{r-x-1}$ yields $(r+1, 1, 2, \ldots, x, n, n-1, \ldots, r+2, x+1, x+2, \ldots, r)$. Then executing L yields $(1, 2, \ldots, x, n, n-1, \ldots, r+2, x+1, x+2, \ldots, r, r+1)$ which is $P_{r+1,n}$. Since $r = 2^k - x$, $r + 1 = 2^k - (x-1)$. Therefore, the starting position of $U(P_{r+1,n})$ in $P_{r+1,n}$ should be $(x-1) + 2 = x + 1$. In $P_{r+1,n}$ the starting position of $U(P_{r+1,n})$ is in fact $x + 1$. Hence by mathematical induction Lemma 1 holds for all values of r.

Lemma 2. *The number of moves required to obtain $P_{r+1,n}$ from $P_{r,n}$ is (a) $2n - 2r$ if r$= 2^k$ for some k. (b) $r - 2^k + 2n - 2$ otherwise.*

Proof. Case (a): $r = 2^k$ for some k.
According to Lemma 1 the starting position of $U(P_{r,n})$ in $P_{r,n}$ is 2. Therefore, $P_{r,n}$ is $(1, n, n-1, \ldots, r+1, 2, 3, \ldots, r)$. Executing $(EL)^{n-r}$ yields $P_{r+1,n}$. Therefore, number of moves to obtain $P_{r+1,n}$ from $P_{r,n}$ when r is in the form of 2^k is $2 * (n - r) = 2n - 2r$.
Case (b): $2^{k-1} < r < 2^k$.

Let us suppose $r = 2^k - x$ where $2^{k-1} < r < 2^k$. According to Lemma 1 the starting position of $U(P_{r,n})$ in $P_{r,n}$ is $x+2$. Therefore, $P_{r,n}$ is $(1, 2, \ldots, x+1, n, n-1, \ldots, r+1, x+2, x+3, \ldots, r)$. Executing $L^{2^k - r}$ i.e. L^x yields $(x+1, n, n-1, \ldots, r+1, x+2, x+3, \ldots, r, 1, 2, \ldots, x)$. Then executing $(EL)^{n-r-1}$ yields $(x+1, r+1, x+2, \ldots, r, 1, 2, \ldots, x, n, n-1, \ldots, r+2)$. Then executing R yields $(r+1, x+2, \ldots, r, 1, 2, \ldots, x, n, n-1, \ldots, r+2, x+1)$. Then executing $(EL)^{2r-2^k-1}$ i.e. $(EL)^{r-x-1}$ yields $(r+1, 1, 2, \ldots, x, n, n-1, \ldots, r+2, x+1, x+2, \ldots, r)$.Then executing L yields $P_{r+1,n}$. Therefore, number of moves to obtain $P_{r+1,n}$ from $P_{r,n}$ when r is not in the form of 2^k is $2^k - r + (2*(n-r-1)) + 1 + (2*(2r-2^k-1)) + 1$ $= r - 2^k + 2n - 2$.

3 Analysis

Lemma 3. *The number of moves required to obtain $P_{3,n}$ from R_n is 1 more than the number of moves required to obtain $P_{3,n-1}$ from R_{n-1}.*

Proof. $P_{3,n}$ is obtained from R_n by executing $n-2$ L moves and an E move, i.e. a total of $n-1$ moves. $P_{3,n-1}$ is obtained from R_{n-1} by executing $n-3$ L moves and an E move, in a total of $n-2$ moves. Therefore, the number of moves required to obtain $P_{3,n}$ from R_n is 1 more than the number of moves required to obtain $P_{3,n-1}$ from R_{n-1}.

Lemma 4. *The number of moves required to obtain $P_{r+1,n}$ from $P_{r,n}$ is 2 more than the number of moves required to obtain $P_{r+1,n-1}$ from $P_{r,n-1}$ $\forall r \in (3, \ldots, n-2)$.*

Proof. (a) According to Lemma 2, if $r = 2^k$ for some k, the number of moves required to obtain $P_{r+1,n}$ from $P_{r,n}$ is $2n - 2r$ where as the number of moves required to obtain $P_{r+1,n-1}$ from $P_{r,n-1}$ is $2(n-1) - 2r = 2n - 2r - 2$. Therefore, the number of moves required to obtain $P_{r+1,n}$ from $P_{r,n}$ is 2 more than number of moves required to obtain $P_{r+1,n-1}$ from $P_{r,n-1}$.
(b) According to Lemma 2, if $2^{k-1} < r < 2^k$ for some k, the number of moves required to obtain $P_{r+1,n}$ from $P_{r,n}$ is $r - 2^k + 2n - 2$ where as the number of moves required to obtain $P_{r+1,n-1}$ from $P_{r,n-1}$ is $r - 2^k + 2(n-1) - 2 = r - 2^k + 2(n) - 4$. Therefore, the number of moves required to obtain $P_{r+1,n}$ from $P_{r,n}$ is 2 more than number of moves required to obtain $P_{r+1,n-1}$ from $P_{r,n-1}$.
From (a) and (b) it follows that Lemma 4 holds for all $r \in (3, 4, \ldots, n-2)$.

Lemma 5. *If $x = 2^k + 1$ for some k then $f(x) = 2x - 5$.*

Proof. Sorting of R_{x-1} involves $(P_{3,x-1}, P_{4,x-1}, \ldots,\ P_{x-1,x-1})$ as intermediate permutations. According to Lemma 3, the number of moves required to obtain $P_{3,x}$ from R_x is 1 more than the number of moves required to obtain $P_{3,x-1}$ from R_{x-1}. According to Lemma 4, the number of moves required to obtain $P_{r+1,x}$ from $P_{r,x}$ is 2 more than number of moves require to obtain $P_{r+1,x-1}$ from $P_{r,x-1}$ $\forall r \in (3, 4, \ldots, x-2)$. Since, $x = 2^k + 1, x - 1 = 2^k$. According to Lemma 2 the number of steps required to obtain $P_{x,x}$ i.e. I_x from $P_{x-1,x}$ is $2x - 2(x-1) = 2$. Therefore, number of additional moves required for sorting R_x when compared to R_{x-1} when $x = 2^k + 1$ for some k is $f(x) = 1 + 2(x-4) + 2 = 2x - 5$.

Lemma 6. *If $x = 2^k + 2$ for some k, $f(x) = 3x - 6$.*

Proof. Sorting of R_{x-1} involves $(P_{3,x-1}, P_{4,x-1}, \dots, P_{x-1,x-1})$ as intermediate permutations. According to Lemma 3, the number of moves required to obtain $P_{3,x}$ from R_x is 1 more than the number of moves required to obtain $P_{3,x-1}$ from R_{x-1}. According to Lemma 4, the number of moves required to obtain $P_{r+1,x}$ from $P_{r,x}$ is 2 more than number of moves require to obtain $P_{r+1,x-1}$ from $P_{r,x-1}$ $\forall r \in (3, 4, \dots, x-2)$. Since, $x = 2^k + 2, x - 1 = 2^k + 1 = 2^{k+1} - (2^k - 1) = 2^{k+1} - (x-3)$. According to Lemma 2, the number of steps required to obtain $P_{x,x}$ i.e I_x from $P_{x-1,x}$ is $(x-1) - 2^{k+1} + 2x - 2 = (x-1) - 2x + 4 + 2x - 2 = x + 1$. Therefore, number of additional moves required for sorting R_x when compared to R_{x-1} when $x = 2^k + 2$ for some k is $f(x) = 1 + 2(x-4) + (x+1) = 3x - 6$.

Lemma 7. *If x not in the form $2^k + 1$ or $2^k + 2$ then $f(x) = f(x-1) + 5$.*

Proof. Recall, $f(x)$ gives us number of additional moves required to sort R_x with LE when compared to R_{x-1}. From Lemma 3 and Lemma 4, we can say that difference between number of moves required to obtain $P_{x-2,x-1}$ from R_{x-1} and the number of moves required to obtain $P_{x-2,x-2}$ i.e. I_{x-2} from R_{x-2} is same as the difference between number of moves required to obtain $P_{x-2,x}$ from R_x and the number of moves required to obtain $P_{x-2,x-1}$ from R_{x-1}
(a) According to Lemma 4, the number of moves required to obtain $P_{x-1,x}$ from $P_{x-2,x}$ is 2 more than number of moves require to obtain I_{x-1} from $P_{x-2,x-1}$.
(b)Let z be the number of moves require to obtain I_{x-1} from $P_{x-2,x-1}$. Since $x-1$ cannot be of the form 2^k for some k, according to Lemma 2 $z = (x-2) - 2^k + 2(x-1) - 2 = x - 2^k + 2x - 6$. Similarly, the number of moves require to obtain I_x from $P_{x-1,x}$ is $(x-1) - 2^k + 2(x) - 2 = x - 2^k + 2x - 3 = z + 3$. The number of moves require to obtain I_x from $P_{x-1,x}$ is 3 more than the number of moves require to obtain I_{x-1} from $P_{x-2,x-1}$.
Therefore, from (a) and (b), $f(x) = f(x-1) + 2 + 3 = f(x-1) + 5$.

Theorem 1. *An upper bound for number of moves required to sort R_n with LE is $\frac{11}{6}n^2$.*

Proof. According to Lemma 5, the value of $f(x)$ when $x = 2^k + 1$ for some k is $2x - 5$.
Therefore, for some k,
$f(2^k + 1) = (2 * (2^k + 1)) - 5 = 2^{k+1} - 3$
According to Lemma 6, the value of $f(x)$ when $x = 2^k + 2$ for some k is $3x - 6$.
$f(2^k + 2) = (3 * (2^k + 2)) - 6 = 3 * 2^k$
According to the Lemma 7,

$$
\begin{aligned}
f(2^k+3) &= f(2^k+3)+5 \\
f(2^k+3) &= (3*(2^k))+5 \\
\text{Similarly, } f(2^k+4) &= (3*(2^k))+5+5 \\
f(2^k+5) &= (3*(2^k))+5+5+5 \\
&\vdots \\
f(2^k+2^k) &= (3*(2^k))+(5+5+\ldots+(2^k-2)times) \\
\text{Let } A(k) &= f(2^k+3)+f(2^k+4)+\ldots+f(2^k+2^k) \\
&= (3*2^k*(2^k-2))+(5+10+15+\ldots+(2^k-2)terms) \\
&= (3*2^k*(2^k-2))+\frac{1}{2}(5*(2^k-2)*(2^k-1)) \\
&= \frac{11}{2}2^{2k}-\frac{27}{2}2^k+5 \\
\text{Let } B(k) &= f(2^k+1)+f(2^k+2)+A(k) \\
&= f(2^k+1)+f(2^k+2)+f(2^k+3)+\ldots+f(2^k+2^k)
\end{aligned}
$$

From Lemma 5 and Lemma 6,

$$
\begin{aligned}
B(k) &= 2^{k+1}-3+(3*2^k)+\frac{11}{2}2^{2k}-\frac{27}{2}2^k+5 \\
&= \frac{11}{2}2^{2k}-\frac{17}{2}2^k+2 \\
B(\log_2(\left\lceil\frac{n}{2}\right\rceil)) &= f(\left\lceil\frac{n}{2}\right\rceil+1)+f(\left\lceil\frac{n}{2}\right\rceil+2)+\ldots+f(\left\lceil\frac{n}{2}\right\rceil+\left\lceil\frac{n}{2}\right\rceil)
\end{aligned}
$$

Therefore, for $M(n)$ the total number of moves to sort R_n we obtain the following recurrence relation.

$$
\begin{aligned}
M(n) &\leq M(\left\lceil\frac{n}{2}\right\rceil)+B(log_2(\left\lceil\frac{n}{2}\right\rceil)) \\
&= \sum_{k=1}^{\log_2 n} B(\log_2(\left\lceil\frac{n}{2^k}\right\rceil)) \\
&= \sum_{k=1}^{\log_2 n} \frac{11}{2}2^{2\log_2(\lceil\frac{n}{2^k}\rceil)}-\frac{17}{2}2^{\log_2(\lceil\frac{n}{2^k}\rceil)}+2 \\
&\text{(Ignoring the lower order terms)} \\
&\leq \sum_{k=1}^{\log_2 n} \frac{11}{2}*2^{2\log_2(\lceil\frac{n}{2^k}\rceil)} \\
&\leq \sum_{k=1}^{\log_2 n} \frac{11}{2}*2^{2\log_2(\frac{n}{2^k}+1)}
\end{aligned}
$$

$$= \sum_{k=1}^{\log_2 n} \frac{11}{2} * (\frac{n}{2^k} + 1)^2$$

(Ignoring the lower order terms)

$$\approx \sum_{k=1}^{\log_2 n} \frac{11}{2} * (\frac{n^2}{2^{2k}})$$

$$\leq \sum_{k=1}^{\infty} \frac{11}{2} * (\frac{n^2}{2^{2k}})$$

$$= \frac{11}{2} * \frac{n^2}{3}$$

$$= \frac{11}{6} * n^2$$

Therefore, an upper bound for number of moves required for R_n with LE is $\frac{11n^2}{6}$.

4 Optimum Algorithm

In this section we design an algorithm, Algorithm LE1 that sorts R_n in optimum number of moves for $n = 1...11$. We claim that Algorithm LE1 indeed sorts R_n in optimum number of moves. We define permutation $V_{r,n} \in S_n$ as follows. If n is even $V_{r,n}$ is divided into 4 sublists. Sublist $(r, r-1, \ldots, \frac{n}{2}+1)$ is denoted by $F_{1,r}(\pi)$. Sublist $(r+1, r+2, \ldots, n)$ is denoted by $F_{2,r+1}(\pi)$.Sublist $(r-\frac{n}{2}, r-\frac{n}{2}-1, \ldots, 2, 1)$ is denoted by $F_{3,r-\frac{n}{2}}(\pi)$. Sublist $(r-\frac{n}{2}+1, r-\frac{n}{2}+2, \ldots, \frac{n}{2})$ is denoted by $F_{4,r-\frac{n}{2}+1}(\pi)$. Then $V_{r,n}$ is obtained by concatenating sublists $F_{1,r}, F_{2,r+1}, F_{3,r-\frac{n}{2}}, F_{4,r-\frac{n}{2}+1}$. If n is odd $V_{r,n}$ is divided into 5 blocks. Sublist $(r, r-1, \ldots, \frac{n+1}{2}, \frac{n-1}{2})$ is denoted by $G_{1,r}(\pi)$. Sublist $(r+1, r+2, \ldots, n-2)$ is denoted by $G_{2,r+1}(\pi)$. Sublist $(r-\frac{n-1}{2}, r-\frac{n-1}{2}-1, \ldots, 2, 1)$ is denoted by $G_{3,r-\frac{n-1}{2}}(\pi)$. Sublist $(n-1, n)$ is denoted by $G_{4,n-1}(\pi)$. Sublist $(r-\frac{n-1}{2}+1, r-\frac{n-1}{2}+2, \ldots, \frac{n-1}{2}-1)$ is denoted by $G_{5,r-\frac{n-1}{2}+1}(\pi)$. Then $V_{r,n}$ is obtained by concatenating sublists $G_{1,r}, G_{2,r+1}, G_{3,r-\frac{n-1}{2}}, G_{4,n-1}, G_{5,r-\frac{n-1}{2}+1}$. Let $(MS)_{1,i,n}$ denotes execution of move sequence $(EL)^i$ followed by $L^{\frac{n}{2}-i}$ followed by $(EL)^i$ followed by $L^{\frac{n}{2}-i}$. Let $(MS)_{2,i,n}$ denote execution of move sequence $(EL)^i$ followed by $L^{\frac{n-1}{2}-i}$ followed by $(EL)^{i+1}$ followed by $L^{\frac{n-1}{2}-i}$.

4.1 Analysis

Lemma 8. *If n is even executing $(MS)_{1,r-\frac{n}{2}-1,n}$ transforms $V_{r,n}$ to $V_{r-1,n}$.*

Proof. According to definition if n is even $V_{r,n}$ is $(r, r-1, \ldots, \frac{n}{2}+1, r+1, r+2, \ldots, n, r-\frac{n}{2}, r-\frac{n}{2}-1, \ldots, 3, 2, 1, r-\frac{n}{2}+1, r-\frac{n}{2}+2, \ldots, \frac{n}{2})$. Executing $(EL)^{r-\frac{n}{2}-1}$ yields $(r, r+1, r+2, \ldots, n, r-\frac{n}{2}, r-\frac{n}{2}-1, \ldots, 2, 1, r-\frac{n}{2}+1, r-\frac{n}{2}+2, \ldots, \frac{n}{2}, r-1, r-2, \ldots, \frac{n}{2}+1)$. Then executing $L^{\frac{n}{2}-(r-\frac{n}{2}-1)}$ i.e. L^{n-r+1} yields $(r-\frac{n}{2}, r-\frac{n}{2}-1, \ldots, 2, 1, r-\frac{n}{2}+1, r-\frac{n}{2}+2, \ldots, \frac{n}{2}, r-1, r-2, \ldots, \frac{n}{2}+$

$1, r, r+1, \ldots, n$. Then executing $(EL)^{r-\frac{n}{2}-1}$ yields $(r-\frac{n}{2}, r-\frac{n}{2}+1, \ldots, \frac{n}{2}, r-1, r-2, \ldots, \frac{n}{2}+1, r, r+1, \ldots, n, r-\frac{n}{2}-1, r-\frac{n}{2}-2, \ldots, 2, 1)$. Then executing $L^{\frac{n}{2}-(r-\frac{n}{2}-1)}$ i.e L^{n-r+1} yields $(r-1, r-2, \ldots, \frac{n}{2}+1, r, r+1, \ldots, n, r-\frac{n}{2}-1, r-\frac{n}{2}-2, \ldots, 2, 1, r-\frac{n}{2}, r-\frac{n}{2}+1, \ldots, \frac{n}{2})$ which equals to concatenation of sublists $F_{1,r-1}(\pi), F_{2,r}(\pi), F_{3,r-\frac{n}{2}-1}(\pi), F_{4,r-\frac{n}{2}}(\pi)$. Therefore obtained permutation is $V_{r-1,n}$.

Algorithm LE1

Input: R_n
Output: I_n
Initialization: $\forall i\ \pi[i] = R_n[i]$. All operations are executed on π.

---

if (n is even) **then**
    for $i \in ((\frac{n}{2}-1), \ldots, 2)$ **do**
        Execute $(MS)_{1,i,n}$
    end for
    Execute EL
    Execute $L^{\frac{n}{2}-1}$
    Execute E
else if (n is odd) **then**
    Execute EL
    Execute L
    for $i \in ((\frac{n-1}{2}-1), \ldots, 2)$ **do**
        Execute $(MS)_{2,i,n}$
    end for
    Execute EL
    Execute $L^{\frac{n-1}{2}-1}$
    Execute $(EL)^2$
end if

Lemma 9. *If n is odd executing $(MS)_{2,r-\frac{n-1}{2},n}$ transforms $V_{r,n}$ to $V_{r-1,n}$.*

Proof. According to definition if n is odd $V_{r,n}$ is $(r, r-1, \ldots, \frac{n+1}{2}, \frac{n-1}{2}, r+1, r+2, \ldots, n-2, r-\frac{n-1}{2}, r-\frac{n-1}{2}-1, \ldots, 2, 1, n-1, n, r+\frac{n-1}{2}+1, r+\frac{n-1}{2}+2, \ldots, \frac{n-1}{2}-1)$. Execution of $(EL)^{r-\frac{n-1}{2}}$ yields $(r, r+1, r+2, \ldots, n-2, r-\frac{n-1}{2}, r-\frac{n-1}{2}-1, \ldots, 2, 1, n-1, n, r+\frac{n-1}{2}+1, r+\frac{n-1}{2}+2, \ldots, \frac{n-1}{2}-1, r-1, \ldots, \frac{n+1}{2}, \frac{n-1}{2})$. Then execution of $L^{\frac{n}{2}-(r-\frac{n-1}{2})}$ i.e. L^{n-r-1} yields $(r-\frac{n-1}{2}, r-\frac{n-1}{2}-1, \ldots, 2, 1, n-1, n, r+\frac{n-1}{2}+1, r+\frac{n-1}{2}+2, \ldots, \frac{n-1}{2}-1, r-1, \ldots, \frac{n+1}{2}, \frac{n-1}{2}, r, r+1, r+2, \ldots, n-2)$. Then execution of $(EL)^{r-\frac{n-1}{2}+1}$ yields $(r-\frac{n-1}{2}, r+\frac{n-1}{2}+1, r+\frac{n-1}{2}+2, \ldots, \frac{n-1}{2}-1, r-1, \ldots, \frac{n+1}{2}, \frac{n-1}{2}, r, r+1, r+2, \ldots, n-2, r-\frac{n-1}{2}-1, \ldots, 2, 1, n-1, n)$. Then execution of $L^{\frac{n}{2}-(r-\frac{n-1}{2})}$ i.e. L^{n-r-1} yields $(r-1, \ldots, \frac{n+1}{2}, \frac{n-1}{2}, r, r+1, r+2, \ldots, n-2, r-\frac{n-1}{2}-1, \ldots, 2, 1, n-1, n, r-\frac{n-1}{2}, r+\frac{n-1}{2}+1, r+\frac{n-1}{2}+2, \ldots, \frac{n-1}{2}-1)$ which equals to concatenation of sublists

$G_{1,r-1}(\pi), G_{2,r}(\pi), G_{3,r-\frac{n-1}{2}-1}(\pi), G_{4,n-1}(\pi), G_{5,r-\frac{n-1}{2}}(\pi)$. Therefore obtained permutation is $V_{r-1,n}$.

Lemma 10. *Algorithm $LE1$ is correct.*

Proof. Case (a): n is even
According to definition $V_{r,n} = R_n$ for $r = n$. According to Lemma 8 if n is even executing $(MS)_{1,r-\frac{n}{2}-1,n}$ transforms $V_{r,n}$ to $V_{r-1,n}$. Therefore after executing $(MS)_{1,r-\frac{n}{2}-1,n}$ for r ranges from n to $\frac{n}{2}+3$ yields $V_{\frac{n}{2}+2,n}$ which equals $(\frac{n}{2}+2, \frac{n}{2}+1, \frac{n}{2}+3, \frac{n}{2}+4, \ldots, n, 2, 1, 3, 4, \ldots, \frac{n}{2})$. Then executing (EL) yields $(\frac{n}{2}+2, \frac{n}{2}+3, \frac{n}{2}+4, \ldots, n, 2, 1, 3, 4, \ldots, \frac{n}{2}, \frac{n}{2}+1)$. Then executing $L^{\frac{n}{2}-1}$ yields $(2, 1, 3, 4, \ldots, \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3, \frac{n}{2}+4, \ldots, n)$ Then executing E yields $(1, 2, 3, 4, \ldots, \frac{n}{2}, \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3, \frac{n}{2}+4, \ldots, n)$ which is I_n.
Case (b): n is odd
Execution of (EL) on R_n yields $(n-1, n-2, \ldots, 1, n)$. Then execution of L yields $(n-2, n-3, \ldots, 1, n, n-1)$ which is equal to $V_{n-2,n}$ when n is odd. According to Lemma 9 if n is odd executing $(MS)_{2,r-\frac{n-1}{2},n}$ transforms $V_{r,n}$ to $V_{r-1,n}$. Therefore after executing $(MS)_{2,r-\frac{n-1}{2},n}$ for r ranges from $n-2$ to $\frac{n-1}{2}+2$ yields $V_{\frac{n-1}{2}+1,n}$ which equals $(\frac{n-1}{2}+1, \frac{n-1}{2}, \frac{n-1}{2}+2, \ldots, n-2, 1, n-1, n, 2, 3, \ldots, \frac{n-1}{2}-1)$. Executing (EL) yields $(\frac{n-1}{2}+1, \frac{n-1}{2}+2, \ldots, n-2, 1, n-1, n, 2, 3, \ldots, \frac{n-1}{2}-1, \frac{n-1}{2})$. Then executing $L^{\frac{n-1}{2}-1}$ yields $(1, n-1, n, 2, 3, \ldots, \frac{n-1}{2}-1, \frac{n-1}{2}, \frac{n-1}{2}+1, \frac{n-1}{2}+2, \ldots, n-2)$. Then executing $(EL)^2$ yields $(1, 2, 3, \ldots, \frac{n-1}{2}-1, \frac{n-1}{2}, \frac{n-1}{2}+1, \frac{n-1}{2}+2, \ldots, n-2, n-1, n)$ which is I_n.

Theorem 2. *The number of moves required to sort R_n with LE is (a)$\frac{3n^2}{4} - 2n$ if n is even, (b)$\frac{3n^2-10n+15}{4}$ if n is odd.*

Proof. Case (a): n is even
Initially $(MS)_{1,i,n}$ is executed for i ranges from $\frac{n}{2}-1$ to 2. Recall, execution of $(MS)_{1,i,n}$ involves execution of $(EL)^i$ followed by $L^{\frac{n}{2}-i}$ followed by execution of $(EL)^i$ and then execution of $L^{\frac{n}{2}-i}$. Therefore, number of moves involved in $(MS)_{1,i,n}$ is $2i + \frac{n}{2} - i + 2i + \frac{n}{2} - i = n + 2i$. Hence, number of moves involved in execution of $(MS)_{1,i,n}$ where i ranges from $\frac{n}{2}-1$ to 2 is $\sum_{i=2}^{\frac{n}{2}-1}(n+2i) = n*(\frac{n}{2}-2) + 2*\sum_{i=2}^{\frac{n}{2}-1} i = \frac{3n^2}{4} - \frac{5n}{2} - 2$. Then execution of (EL) followed by $L^{\frac{n}{2}-1}$ followed by E yields I_n. Therefore, total number of moves required to sort R_n with LE when n is even is $\frac{3n^2}{4} - \frac{5n}{2} - 2 + 2 + \frac{n}{2} - 1 + 1 = \frac{3n^2}{4} - 2n$.
Case (b): n is odd
Initially EL followed by L move is executed on R_n. Then $(MS)_{2,i,n}$ is executed for i ranges from $\frac{n-1}{2}-1$ to 2. And finally execution of (EL) followed by $L^{\frac{n-1}{2}-1}$ followed by $(EL)^2$ sorts R_n. Recall, execution of $(MS)_{2,i,n}$ involves execution of $(EL)^i$ followed by $L^{\frac{n-1}{2}-i}$ followed by execution of $(EL)^{i+1}$ and then execution of $L^{\frac{n-1}{2}-i}$. Therefore, number of moves involved in $(MS)_{2,i,n}$ is $2i + \frac{n-1}{2} - i + 2(i+1) + \frac{n-1}{2} - i = n + 2i + 1$. Hence, number of moves involved in execution of $(MS)_{2,i,n}$ where i ranges from $\frac{n-1}{2}-1$ to 2 is $\sum_{i=2}^{\frac{n-1}{2}-1}(n+2i+1) = (n+$

$1) * (\frac{n-1}{2} - 2) + 2 * \sum_{i=2}^{\frac{n-1}{2}-1} i = \frac{3n^2}{4} - 3n - \frac{15}{4}$. Therefore, total number of moves required to sort R_n with LE when n is odd is $2+1+\frac{3n^2}{4}-3n-\frac{15}{4}+2+\frac{n-1}{2}-1+4 = \frac{3n^2-10n+15}{4}$.

5 Conclusions

We derived a novel upper bound for LE operation. This is operation has the fewest generators that are needed to generate S_n. A variation of LE is the LRE operation where an additional right rotate is allowed. For LRE operation we conjecture that the optimum number of moves is $(n^2 - n - 4)/2 \; \forall \; n > 3$.

References

1. Kuppili1 S. S., Chitturi1, B., Srinath T.: An Upper Bound for Sorting R_n with LE. ICACDS 2019.
2. The On-Line Encyclopedia of Integer Sequences: oeis.org.
3. Jerrum, M. R.: The complexity of finding minimum-length generator sequences. Theoretical Computer Science, 36, 265-289 (1985).
4. Akers, S. B., Krishnamurthy, B.: A group-theoretic model for symmetric interconnection networks. IEEE Transactions on Computers, 38(4):555–566 (1989).
5. Chitturi, B., Fahle, W., Meng, Z., Morales, L., Shields, C. O., Sudborough H.: An (18/11) n upper bound for sorting by prefix reversals. Theoretical Computer Science, 410(36), 3372-3390 (2009).
6. Chen, T., Skiena, S. S.: Sorting with fixed-length reversals. Discrete Applied Mathematics, 71(1-3), 269-295 (1996).
7. Feng, X., Chitturi,B., Sudborough, H.: Sorting circular permutations by bounded transpositions. In Advances in Computational Biology, (pp. 725-736). Springer, New York, NY (2010).
8. Chitturi, B., Sudborough, H., Voit, W., Feng, X.: Adjacent swaps on strings. In International Computing and Combinatorics Conference, (pp. 299-308). Springer, Berlin, Heidelberg (2008).
9. Lakshmivarahan, S., Jho, J-S., and Dhall, S. K.: Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey. Parallel Computing, 19:361–407, (1993).
10. Chitturi, B.: Perturbed Layered Graphs. ICACCP (2019).
11. Chitturi, B., Balachander, S., Satheesh S., and Puthiyoppil K. Layered Graphs: Applications and Algorithms. Algorithms 2018, 11(7), 93.
12. Mokhtar, H.: A few families of Cayley graphs and their efficiency as communication networks. Bulletin of the Australian Mathematical Society 95, no. 3 : 518-520 (2017).
13. Zhang, T., and Gennian G.: Improved lower bounds on the degreediameter problem. Journal of Algebraic Combinatorics, 1-12 (2018).
14. Chitturi,B., and Das, P.:Sorting permutations with transpositions in $O(n^3)$ amortized time. Theoretical Computer Science, (2018).
15. Erskine, G., and James T.: Large Cayley graphs of small diameter. Discrete Applied Mathematics (2018).
16. Gostevsky, D.A. and Konstantinova, E.V.: Greedy cycles in the star graphs. Discrete mathematics and mathematical cybernetics, 15(0): 205-213 (2018).

17. Fertin, G., Labarre, A., Rusu, I., Vialette, S. and Tannier, E.: Combinatorics of genome rearrangements. MIT press (2009).
18. Chitturi, B.: A note on complexity of genetic mutations. Discrete Mathematics, Algorithms and Applications, 3.03: 269-286 (2011).