

Polynomial Block Lengths Are Necessary and Sufficient to Beat Insertion Sort

Nikolai Chukhin^{*1}, Alexander S. Kulikov^{†1}, Maksim Levitskii^{‡1,2}, and Ivan Mihajlin^{§1}

¹JetBrains Research

²Neapolis University Pafos

Abstract

Insertion sort is a basic algorithm for sorting permutations: it repeatedly cuts out one element and inserts it at its place among the elements sorted so far. It sorts every permutation of length n with at most $n - 1$ such moves, and some permutations cannot be sorted with fewer moves (Heath and Vergara, 1998). A move that cuts out a block of any length is called a transposition, and the number of transpositions that suffices to sort every permutation lies between $n/2$ and $2n/3$ for $n \geq 9$ (Bafna and Pevzner, 1998; Eriksson, Eriksson, Karlander, Svensson, and Wästlund, 2001). We ask how long the cut blocks must be to beat insertion sort by a constant factor.

We show that cut blocks of polynomial length are necessary and sufficient. If the cut blocks have length $n^{o(1)}$, then some permutations need $n - o(n)$ moves, whereas for any $\alpha > 0$, $(1 - \varepsilon_\alpha)n$ moves with cut blocks of length n^α suffice for all large enough n .

The lower bound follows from a new upper bound on the number of permutations at a given distance from the identity. The upper bound follows from a recursive reduction of the exponent, starting from the known bound for unrestricted transpositions.

2012 ACM Subject Classification: Mathematics of computing — Permutations and combinations; Theory of computation — Sorting and searching.

Keywords and phrases: sorting by transpositions, transposition diameter, insertion sort.

^{*}buyolitsez1951@gmail.com

[†]alexander.s.kulikov@gmail.com

[‡]max.levitsky.5@gmail.com

[§]ivmihajlin@gmail.com

1 Transpositions with Bounded Block Length

We call the following operation a *transposition*: cut a block out of a permutation and insert it elsewhere [1]. Transpositions are a basic operation in genome rearrangements, where they model the relocation of a segment of a chromosome [1, 6], and in edit distances with moves [9, 3]. The *transposition distance* of a permutation is the minimum number of transpositions that sort it, and the *transposition diameter* $D(n)$ is the maximum transposition distance of a permutation of length n .

Sorting by transpositions. The transposition diameter turned out to be surprisingly hard to determine. Bafna and Pevzner [1] introduced the problem of computing it, gave a 1.5-approximation algorithm for it, and showed that $\lfloor n/2 \rfloor \leq D(n) \leq 3n/4$. [5] showed that $D(n) \leq \lfloor (2n-2)/3 \rfloor$ for $n \geq 9$, and these bounds have since been improved only by additive constants [4, 10]. Thus, even the leading constant of $D(n)$ has remained unknown for more than two decades, and the exact value $D(16) = 9$ was established only in 2026, by a computer-assisted proof [10]. Moreover, Bulteau, Fertin and Rusu [2] showed that computing the transposition distance between two permutations is NP-hard.

Bounded block lengths. Heath and Vergara [7] proposed bounding the length of the *cut block*: a move cuts out a block of at most k elements and inserts it anywhere. Let $D_{\leq k}(n)$ be the corresponding diameter, that is, the maximum, over all permutations of length n , of the minimum number of such moves that sort the permutation.¹ For $k = 1$, a move reinserts only a single element, and [7] showed that the minimum number of moves that sort a permutation π is $n - \text{LIS}(\pi)$, where $\text{LIS}(\pi)$ is the length of a longest increasing subsequence of π . Hence, $D_{\leq 1}(n) = n - 1$, and insertion sort is optimal for $k = 1$. Since $D_{\leq k}(n)$ does not increase with k , we have

$$\lfloor n/2 \rfloor \leq D(n) \leq D_{\leq k}(n) \leq n - 1.$$

To the best of our knowledge, nothing more was known about $D_{\leq k}(n)$ for $1 < k < n$. This raises a natural question: how long must the cut blocks be to beat insertion sort by a constant factor?

1.1 Our Results

We show that blocks of polynomial length are necessary and sufficient to beat insertion sort by a constant factor.

Theorem 1. *Let $1 \leq k \leq n$.*

- *If $k = n^{o(1)}$, then $D_{\leq k}(n) = n - o(n)$.*
- *If $k \geq n^\alpha$ for a constant $\alpha > 0$, then $D_{\leq k}(n) \leq (1 - \varepsilon_\alpha)n + o(n)$ for a constant $\varepsilon_\alpha > 0$.*

Interestingly, for every $k \geq 2$, the arguments in [1, 7] cannot give a lower bound better than $n/2$, so they do not distinguish moves of two elements from unrestricted transpositions, although $D_{\leq 1}(n) = n - 1$ for moves of one element, whereas $D_{\leq n}(n) = D(n) \leq 2n/3$. One can also consider a naive counting argument that bounds the number of permutations at distance at most T by $O((n^2k)^T)$: there are at most nk choices for the cut block and n choices for the insertion position. For $k = n^{o(1)}$, applying this estimate with $T = D_{\leq k}(n)$ gives only $D_{\leq k}(n) \geq n/2 - o(n)$.

Our lower bound also counts sequences of moves, but it avoids counting many sequences that sort the same permutation. We show that every sequence of moves can be reordered into a *normal form* without changing the resulting permutation, and we count only sequences in normal form, which gives the following bound.

¹A different model involving bounds not only on the block length but also on the shift distance was studied by Heath and Vergara [8], who showed that the diameter is quadratic in this case.

Theorem 2. For $n \geq 2$, $1 \leq k \leq n$, and $0 \leq T \leq n$,

$$|\{\pi \in S_n : d_{\leq k}(\pi) \leq T\}| \leq 2^{O(n)}(nk^2)^T.$$

Compared with the bound $O((n^2k)^T)$ of the direct count, we save a factor of n/k per move, at the price of a factor $2^{O(n)}$. For $k = n^{o(1)}$, it gives $D_{\leq k}(n) = n - o(n)$.

Our upper bound comes from a recursion on k . To sort with cut blocks of length at most $n^{\alpha'}$, we repeatedly gather a small set of elements and sort them using the bound for the larger exponent $\alpha > \alpha'$.

Theorem 3. Let $0 < \alpha \leq 1$ and $0 < \varepsilon < 1$. Suppose that

$$D_{\leq n^\alpha}(n) \leq (1 - \varepsilon)n + o(n).$$

Then, for $\frac{\alpha}{1+\alpha} < \alpha' < \alpha$, there exists $\varepsilon' > 0$ such that

$$D_{\leq n^{\alpha'}}(n) \leq (1 - \varepsilon')n + o(n).$$

Starting from the bound $D(n) \leq 2n/3$ for $\alpha = 1$ [5], the recursion reaches every constant $\alpha > 0$.

2 Preliminaries

Throughout the paper, $[n] = \{1, \dots, n\}$, and S_n denotes the set of all permutations of $[n]$.

Transposition diameter problem. A *cut* of a permutation is a position between two of its consecutive elements, before its first element, or after its last element. A transposition cuts out a nonempty contiguous block of a permutation, the *cut block*, and inserts it at a cut of the remaining sequence other than the one where the block was, keeping the order of the elements in the block. This is the original definition of Bafna and Pevzner [1]; it is often stated equivalently as swapping two adjacent nonempty blocks B and C while keeping the other elements in place [2]. We also call transpositions *moves*. Let $d(\pi)$ be the minimum number of moves that *sort* π , that is, transform it into the identity permutation, and let

$$D(n) = \max_{\pi \in S_n} d(\pi).$$

Bounded-length transpositions. In the bounded-length version of the problem, the cut block has length at most k for some real $k \geq 1$; equivalently, $\min\{|B|, |C|\} \leq k$. We call such moves *allowed*, and define $d_{\leq k}(\pi)$ and $D_{\leq k}(n)$ as above, using only allowed moves. Moving single elements one by one, as in insertion sort, sorts every permutation in at most $n - 1$ allowed moves, so $D_{\leq k}(n) \leq n - 1$.

3 Lower Bound

The overall idea of the proof is to bound the number of permutations that can be obtained within a bounded number of moves. Hence, as long as their number is less than $n!$, we can obtain a lower bound on the number of moves required.

Theorem 2. For $n \geq 2$, $1 \leq k \leq n$, and $0 \leq T \leq n$,

$$|\{\pi \in S_n : d_{\leq k}(\pi) \leq T\}| \leq 2^{O(n)}(nk^2)^T.$$

Applying the theorem to $T = D_{\leq k}(n)$ gives the lower bound.

Corollary 4. For $n \geq 2$ and $1 \leq k \leq n$,

$$D_{\leq k}(n) \geq \frac{n \log n - O(n)}{\log(nk^2)}.$$

Proof. Let $T = D_{\leq k}(n)$. Then $T \leq n - 1$, and all $n!$ permutations $\pi \in S_n$ satisfy $d_{\leq k}(\pi) \leq T$ by definition. Hence, by Theorem 2, $n! \leq 2^{O(n)}(nk^2)^T$, and taking logarithms gives

$$n \log n - O(n) \leq \log(n!) \leq D_{\leq k}(n) \log(nk^2) + O(n).$$

Rearranging finishes the proof. $\square$

To prove Theorem 2, we bound the number of permutations reachable from the identity in exactly T allowed moves.

3.1 Encoding the Moves

Instead of encoding a cut c by its position in the permutation, we encode it by the pair $(\text{left}(c), s(c))$, where $\text{left}(c)$ is the nearest marked element to the left of c , and the *shift* $s(c)$ is the number of elements between $\text{left}(c)$ and c . To make sure that there is a marked element to the left of each position, we prepend the element 0 to the permutation and always mark it. For now, the other marked elements are arbitrary; we choose them later in Lemma 6.

Consider a history of T moves. We encode it one move at a time and after we encoded a move $v \in [T]$, we apply it, and proceed to move $v + 1$. Move v is given by three cuts L_v, R_v, I_v : the cut block lies between L_v and R_v , and it is inserted at the cut I_v , which lies outside the cut block. Let b_v be the number of marked elements in the cut block of move v . Move v is encoded by the marked elements to the left of its cuts L_v and I_v , the number b_v , and the shifts of its three cuts:

$$\langle \text{left}(L_v), \text{left}(I_v), b_v, s(L_v), s(R_v), s(I_v) \rangle.$$

Given the current permutation, those numbers determine the move, since $\text{left}(R_v)$ is the b_v -th marked element after $\text{left}(L_v)$, and each cut is determined by its marked element and its shift.

3.2 Reordering the Moves

Many different histories produce the same permutation at the end, and we now normalize the history to remove most of this redundancy. The *support* of a move $v \in [T]$ is the set of marked elements that its encoding uses or that it moves:

$$\text{supp}(v) = \{\text{left}(L_v), \text{left}(I_v)\} \cup \{\text{marked elements in the cut block of } v\}.$$

If two consecutive moves have disjoint supports, they can be interchanged, since neither move changes the marked elements or the cut data used to specify the other.

We group the moves into levels. For each move v , define its *level* $h(v)$ recursively by

$$h(v) = 1 + \max\{h(u) : u < v \text{ and } \text{supp}(u) \cap \text{supp}(v) \neq \emptyset\},$$

where the maximum over the empty set is zero. Thus, moves of the same level have pairwise disjoint supports, and every move of level $i + 1$ shares a marked element with a move of level i . The next lemma lets us assume that the moves are ordered by level.

Lemma 5. *The moves can be reordered so they appear in nondecreasing order of their h values, without changing the final permutation.*

Proof. Consider two adjacent moves that form an inversion in their h values. Their supports are disjoint, since otherwise the later move would have the larger h value. Hence, the two moves can be interchanged. Repeating this operation eliminates all inversions and preserves the final permutation. $\square$

3.3 Counting the Histories

We now bound the number of permutations reachable in exactly T moves. Let M be the number of marked elements, including 0. Let $b = \sum_{v \in [T]} b_v$ be the total number of marked elements in the cut blocks, and let $A = \sum_{v \in [T]} (s(L_v) + s(R_v) + s(I_v))$ be the *total shift*.

We choose marked elements independently for each history so that M , A , and b are small. The set of marked elements is also part of the encoding; since there are at most 2^n such sets, it is sufficiently small for our purposes.

Lemma 6. *For $T \leq n$, every history of T moves admits a choice of marked elements such that*

$$\frac{A}{k} + Mk + b \leq 6n;$$

in particular, $M \leq 6n/k$, $b \leq 6n$, and $A \leq 6nk$.

Proof. Mark each element of $[n]$ independently with probability $1/k$. The shift of a cut is at least j only if the j elements to its left are unmarked, so its expectation is at most $\sum_{j \geq 1} (1 - 1/k)^j = k - 1$. Since every cut block has at most k elements, linearity of expectation gives, for a fixed history,

$$\mathbb{E}M = 1 + \frac{n}{k} \leq \frac{2n}{k}, \quad \mathbb{E}b \leq T \leq n, \quad \mathbb{E}A \leq 3T(k - 1) \leq 3nk.$$

Hence, $\mathbb{E}[A/k + Mk + b] \leq 6n$, so some choice of the marked elements satisfies $A/k + Mk + b \leq 6n$. $\square$

Fix such a choice of marked elements. By Lemma 5, we may assume that the moves of the history are ordered by level.

Proof of Theorem 2. We first bound the number of permutations reachable in exactly $T \geq 1$ moves. Every such permutation is determined by the marked elements given by Lemma 6, with at most 2^n choices, and by the encodings of the moves of its history ordered by level. We count the parts of these encodings separately, using $T \leq n$, $M \leq 6n/k$, $b \leq 6n$, and $A \leq 6nk$.

Let U_0 be the set of all marked elements, and for $i \geq 1$, let $U_i = \bigcup_{h(v)=i} \text{supp}(v)$ be the set of marked elements used by the moves of level i . Since supports on each level are disjoint and $|\text{supp}(v)| \leq b_v + 2$,

$$\sum_{i \geq 0} |U_i| \leq M + \sum_{v \in [T]} (b_v + 2) = M + b + 2T = O(n).$$

We now show how to encode the pair $(\text{left}(L_v), \text{left}(I_v))$ of a move v of level i efficiently. Let its *witness* F_v be the smallest element of $\text{supp}(v) \cap U_{i-1}$, which is nonempty by the definition of levels. Since $F_v \in \text{supp}(v)$, there are three cases:

- if $F_v = \text{left}(I_v)$, we encode F_v , and $\text{left}(L_v)$;
- if $F_v = \text{left}(L_v)$, we encode F_v , and $\text{left}(I_v)$;
- if F_v is the j -th of the b_v marked elements in the cut block, then $\text{left}(L_v)$ is the j -th marked element before F_v , and we encode F_v, j, b_v and $\text{left}(I_v)$.

We also encode which case holds using two bits. Thus, besides F_v , every move encodes one marked element, with at most M choices, and $O(b_v + 1)$ bits, which is $O(b + T) = O(n)$ bits for all moves. Moves of the same level have disjoint supports, so their witnesses are distinct, and we encode the witnesses of level i together as a subset of U_{i-1} using $O(n)$ bits in total. Hence, the pairs of all moves have at most $2^{O(n)} M^T$ possibilities.

Finally, we encode shifts $s(L_v)$, $s(R_v)$, $s(I_v)$ of all moves $v \in [T]$. They are nonnegative integers with sum $A \leq 6nk$, so, using $(n/T)^T \leq e^n$, there are at most

$$\binom{6nk + 3T}{3T} \leq \left(\frac{e(6nk + 3T)}{3T} \right)^{3T} \leq \left(\frac{3enk}{T} \right)^{3T} = 2^{O(n)} k^{3T}$$

choices.

Observe that now, given an encoding we know T , set of all marked elements U_0 , values of $(\text{left}(L_v), \text{left}(I_v))$ and $s(L_v)$, $s(R_v)$, $s(I_v)$, so we can decode the permutation. Multiplying all bounds, we obtain that the number of permutations reachable in exactly T moves is at most $2^{O(n)} M^T k^{3T} \leq 2^{O(n)} (6n/k)^T k^{3T} = 2^{O(n)} (nk^2)^T$. Summing this bound over history lengths from 0 to T gives at most $(T+1)2^{O(n)} (nk^2)^T = 2^{O(n)} (nk^2)^T$ permutations, which proves the theorem. $\square$

4 Upper Bound

In this section, we prove $D_{\leq n^\alpha}(n) \leq (1 - \varepsilon_\alpha)n + o(n)$ for every $\alpha > 0$. To obtain this bound, we use a recursion that starts with the case $k = n$ and descends to smaller block lengths. The recursive step is captured by the following theorem.

Theorem 3. *Let $0 < \alpha \leq 1$ and $0 < \varepsilon < 1$. Suppose that*

$$D_{\leq n^\alpha}(n) \leq (1 - \varepsilon)n + o(n).$$

Then, for $\frac{\alpha}{1+\alpha} < \alpha' < \alpha$, there exists $\varepsilon' > 0$ such that

$$D_{\leq n^{\alpha'}}(n) \leq (1 - \varepsilon')n + o(n).$$

Applying the theorem repeatedly, starting from $\alpha = 1$ for which $D(n) \leq \frac{2}{3}n + o(n)$ [5], we reach every exponent $\alpha > 0$.

Corollary 7. *For every fixed $\alpha > 0$, there exists $\varepsilon > 0$ such that*

$$D_{\leq n^\alpha}(n) \leq (1 - \varepsilon)n + o(n).$$

We measure how far a permutation is from being sorted by the number of its *strips*.

Definition 8. A *strip* is a maximal contiguous block of the form $x, x+1, \dots, x+w-1$.

An *adjacency* is an element x immediately followed by $x+1$, and a *breakpoint* is a pair of neighboring elements that is not an adjacency; a permutation with r strips has exactly $n - r$ adjacencies, so creating an adjacency reduces the number of strips by one, and a permutation is sorted if and only if it consists of one strip. The next lemma shows that it suffices to reduce the number of strips to sort the permutation cheaply.

Lemma 9. *If a permutation consists of r strips, then it can be sorted in at most $r + n/k$ allowed moves.*

Proof. Split every strip of length w into $\lceil w/k \rceil$ consecutive parts. The total number of parts is at most $\sum_i \lceil w_i/k \rceil \leq r + n/k$, where the sum ranges over the lengths w_i of all strips. Then, just move these parts to the end of the permutation in increasing order of their elements. $\square$

From now on, let α , ε , and α' be as in Theorem 3, and assume its hypothesis $D_{\leq n^\alpha}(n) \leq (1 - \varepsilon)n + o(n)$. We sort a permutation of $[n]$ by allowed moves for $k = \lfloor n^{\alpha'} \rfloor$ and measure progress by the number r of strips of the current permutation.

By Lemma 9, sorting a permutation with r strips costs about one move per strip, so to beat the trivial bound, we have to remove some strips more cheaply. Note that if $r \leq 9n/10$, then Lemma 9 sorts the permutation in $r + n/k \leq 9n/10 + o(n)$ moves. Hence, we assume that $r > 9n/10$. In this case, we keep removing strips until at most $9n/10$ of them remain, using at most $1 - \nu$ moves per removed strip for some constant $\nu > 0$. Then, we reduce r from at most n to some $r_0 \leq 9n/10$ using at most $(1 - \nu)(n - r_0)$ moves, and sort the resulting permutation in $r_0 + n/k$ moves by Lemma 9. Since $n - r_0 \geq n/10$, the total number of moves is at most

$$(1 - \nu)(n - r_0) + r_0 + n/k \leq (1 - \nu/10)n + o(n).$$

From now on $r > 9n/10$. Then, at least $8r/9$ of the strips have length 1, since otherwise more than $r/9$ strips have length at least 2, and the total length exceeds $2(r/9) + 8r/9 = 10r/9 > n$, a contradiction. A natural step is to group the current strips into *windows* of total length at most k , since inside such a window we may move blocks of arbitrary length. We construct the windows taking strips greedily from left to right, i.e., each window is a maximal set of consecutive strips of total length at most k , while a strip of length greater than k forms a window on its own. Since every two consecutive windows have total length greater than k , there are at most $2n/k + 1 = O(n/k)$ of them.

Let E be the number of elements $x \in [n - 1]$ such that x and $x + 1$ lie in different strips of the same window. We reduce the number of strips in two different ways, depending on whether E is large or small. An easy case is when E is big.

Lemma 10. *For all large enough n , every permutation of $[n]$ with $E \geq 3r/4$ can be turned by allowed moves into a permutation with fewer strips, using at most $9/10$ moves per removed strip.*

Proof. We sort the strips inside every window into increasing order. Every window with more than one strip has length at most k , so we can sort it by unrestricted transpositions, hence, in total, we use at most $2r/3 + O(n/k)$ moves [5]. After sorting, we removed at least E strips. Since $E \geq 3r/4$ and $r > 9n/10$, we have $2r/3 \leq 8E/9$ and $n/k = o(E)$, so the number of moves is at most $(8/9 + o(1))E \leq 9E/10$ for large n , which proves the claim. $\square$

So, when E is big we created adjacencies *inside* each window. Then, if E is small, we instead create adjacencies *across* windows. In this case, we gather many strips of length 1 into one contiguous block at the end of the permutation and sort this block by the hypothesis of Theorem 3. The block is much longer than n/k , so that the $O(n/k)$ moves that assemble it are negligible, and shorter than $k^{1/\alpha}$, so that the hypothesis sorts it with cut blocks of length less than k .

So, we select some strips of length 1 that we want to gather at the end of the permutation. However, moving them one by one is not efficient. Instead, inside each window, we move every block that lies between two selected strips to where it belongs: if the block ends with x , we insert it immediately before $x + 1$. Each such move creates an adjacency, so it pays for itself, and afterward the selected strips of the window form one contiguous block, which a single move takes to the end of the permutation. Note that the destinations of these moves must lie in windows without selected strips, otherwise, a moved block could land between two selected strips that are already contiguous and separate them again, or inside a block that we move later, which could then become longer than k . If we only cut blocks from windows with selected strips and only insert them into other windows, then every block is still in its original place when we move it, and every created adjacency survives.

Consider a strip that is not the first in its window, and let x be the last element of the strip to its left. If $x < n$, we call $x + 1$ the *replacement* of the strip, since $x + 1$ should stand in its place, since moving a block that ends with x immediately before $x + 1$ creates an adjacency.

It remains to decide which strips of length 1 to select. After we sort the selected strips at the end of the permutation, we want them to form, ideally, one strip. But we will make them

to be almost a single strip, i.e. with a small number of gaps between them, and we later gather remaining elements to fill the gaps. Hence, we select strips whose values lie in one interval W of m consecutive values. The gathered block then has $\Theta(m)$ elements, so we take m much larger than n/k but with $m^\alpha < k$.

Lemma 11. *Let $m = o(n)$. For all large enough n , if $E < 3r/4$, then some interval W of m consecutive values contains $\Omega(m)$ strips of length 1 whose replacements lie in other windows.*

Proof. Recall that at least $8r/9$ of the strips have length 1. Among these strips, at most one per window, are the first strips in their windows, i.e. $2n/k + 1 = o(n)$. By the definition of E , at most E of them have their replacements in the same window. Thus, at least $8r/9 - E - o(n) \geq 5r/36 - o(n) = \Omega(n)$ strips of length 1 have their replacements in other windows. Now, split $[n]$ into $\lfloor n/m \rfloor$ intervals of m consecutive values. By the pigeonhole principle, one of the intervals contains at least $\Omega(n)/(n/m) = \Omega(m)$ such strips. $\square$

As described above, we cut blocks only from windows with selected strips and insert them only into windows without selected strips. The next lemma shows that we can do this and still keep $\Omega(m)$ of the strips given by Lemma 11: we color the windows red and blue, and select only strips in red windows whose replacements lie in blue windows. Let W be the interval supplied by Lemma 11.

Lemma 12. *The windows can be colored red and blue so that W contains $\Omega(m)$ strips of length 1 whose windows are red and whose replacements lie in blue windows.*

Proof. Color each window independently red or blue with equal probability. For each strip from W , the probability that its window is red and the window containing its replacement is blue is $1/4$. Hence, the expected number of such strips in W is $\Omega(m)/4 = \Omega(m)$, so some coloring has at least $\Omega(m)$ suitable strips. $\square$

We now sort these strips using the hypothesis of Theorem 3.

Lemma 13. *There exists $\nu > 0$ such that, for all large enough n , every permutation of $[n]$ with $E < 3r/4$ can be turned by allowed moves into a permutation with fewer strips, using at most $1 - \nu$ moves per removed strip.*

Proof. Choose θ such that $1 - \alpha' < \theta < \alpha'/\alpha$, and set $m = \lfloor n^\theta \rfloor$, hence $n/k = o(m)$, $m = o(n)$ and $m^\alpha < k$ for large n . By Lemma 12, for some absolute constant $c > 0$, we can color the windows so that some interval W of m consecutive values contains cm strips of length 1 whose windows are red and whose replacements lie in blue windows; we call these strips *selected*. Then, we do the following:

1. In each window that contains selected strips, move the block between every two consecutive selected strips immediately before the replacement of the second of these strips.
2. In each window, the selected strips now form one contiguous block; move this block to the end of the permutation.
3. Sort the cm selected strips of length 1, using the $D_{\leq (cm)^\alpha}(n) \leq (1 - \varepsilon)cm + o(m)$ moves.
4. Fill the gaps between the selected elements with the other elements of W greedily: cut them into maximal blocks of consecutive values that are contiguous in the permutation, split every block longer than k , and insert each block at its place among the selected elements.

Every move in Step 1 takes a block from a red window and inserts it into a blue one. Red windows receive no blocks, so the cut block of every move lies inside one original window and has length at most k ; blue windows contain no selected strips, so nothing in them is moved in Steps 1 and 2, and the adjacencies created in Step 1 remain.

Let m_i be the number of moves in step $i \in [4]$. Each move of Step 1 creates an adjacency. Also, almost each move of Step 4 creates an adjacency, except moves that split every block of length logner than k , but there might be at most $m/k = o(m)$ of such moves. Hence, the procedure creates $m_1 + cm + m_4 - o(m)$ adjacencies. Also, ut uses $m_1 + m_2 + m_3 + m_4$ moves in total. Note that $m_2 = O(n/k) = o(m)$, and, since $(cm)^\alpha < k$, then $m_3 \leq (1 - \varepsilon)cm + o(m)$. Therefore, the number of new adjacencies exceeds the number of moves by at least $\varepsilon cm - o(m)$. Since $m_1 \leq cm$ and $m_4 \leq (1 - c)m + o(m)$, the procedure uses $M \leq 2m + o(m) \leq 3m$ moves and removes at least $M + \varepsilon cm/2$ strips for large n . Hence, it uses at most $M/(M + \varepsilon cm/2)$ moves per removed strip, and since this ratio increases with M and $M \leq 3m$, it is at most

$$\frac{3m}{3m + \varepsilon cm/2} = \frac{1}{1 + \varepsilon c/6} \leq 1 - \frac{\varepsilon c}{12},$$

which proves the lemma with $\nu = \varepsilon c/12$. $\square$

Proof of Theorem 3. Let $\nu > 0$ be the constant from Lemma 13; we may assume that $\nu \leq 1/10$. While $r > 9n/10$, we reduce the number of strips using Lemma 10 when $E \geq 3r/4$, and using Lemma 13 otherwise; in both cases, every removed strip costs at most $1 - \nu$ moves for large n . Hence, by Lemma 9, the claim holds with $\varepsilon' = \nu/10$. $\square$

AI Disclosure

The main parts of the proofs of Theorems 2 and 3 were produced by the AI model GPT 6 Astra. Below, we describe how these results were obtained.

We started by trying to prove lower bounds on $D_{\leq k}(n)$ for constant k with the help of Astra. First, we obtained $D_{\leq 2}(n) \geq \frac{3}{4}n - o(n)$ and $D_{\leq 3}(n) \geq \frac{4}{7}n - o(n)$. In an iterative process, the coefficient of n in the lower bound on $D_{\leq 3}(n)$ then improved to $\frac{4}{7} + \frac{1}{329}$, to $\frac{29}{50}$, and finally to $\frac{3}{5}$. Since we could not exclude that $D_{\leq 3}(n) \leq (1 - \varepsilon)n$ for some $\varepsilon > 0$, we also spent much effort trying to prove such an upper bound. Then, to our surprise, Astra gave a proof that $D_{\leq k}(n) \geq n - o(n)$ for every constant k , which we generalized to all $k = n^{o(1)}$, obtaining Theorem 2. After that, we studied $D_{\leq k}(n)$ for polynomial k . We quickly proved that $D_{\leq n^{3/4}}(n) \leq (1 - \varepsilon)n + o(n)$ for some $\varepsilon > 0$, and this approach then extended to Theorem 3. The authors are fully responsible for the content of this paper.

References

- [1] Vineet Bafna and Pavel A. Pevzner. Sorting by Transpositions. *SIAM J. Discret. Math.*, 11(2):224–240, 1998. doi:10.1137/S089548019528280X.
- [2] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by Transpositions Is Difficult. *SIAM J. Discret. Math.*, 26(3):1148–1180, 2012. doi:10.1137/110851390.
- [3] Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1):2:1–2:19, 2007. doi:10.1145/1219944.1219947.
- [4] Isaac Elias and Tzvika Hartman. A 1.375-Approximation Algorithm for Sorting by Transpositions. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 3(4):369–379, 2006. doi:10.1109/TCBB.2006.44.

- [5] Henrik Eriksson, Kimmo Eriksson, Johan Karlander, Lars J. Svensson, and Johan Wästlund. Sorting a bridge hand. *Discret. Math.*, 241(1-3):289–300, 2001. doi:10.1016/S0012-365X(01)00150-9.
- [6] Guillaume Fertin, Anthony Labarre, Irena Rusu, Eric Tannier, and Stéphane Vialette. *Combinatorics of Genome Rearrangements*. MIT Press, 2009. URL: <https://mitpress.mit.edu/books/combinatorics-genome-rearrangements>.
- [7] Lenwood S. Heath and John Paul C. Vergara. Sorting by Bounded Block-moves. *Discret. Appl. Math.*, 88(1-3):181–206, 1998. doi:10.1016/S0166-218X(98)00072-9.
- [8] Lenwood S. Heath and John Paul C. Vergara. Sorting by Short Block-Moves. *Algorithmica*, 28(3):323–354, 2000. doi:10.1007/S004530010041.
- [9] Dana Shapira and James A. Storer. Edit distance with move operations. *J. Discrete Algorithms*, 5(2):380–392, 2007. doi:10.1016/J.JDA.2005.01.010.
- [10] Luiz A. G. Silva, Luis A. B. Kowada, Noraí R. Rocco, and Maria E. M. T. Walter. Twisted Bracelets for Sorting by Transpositions: the Transposition Diameter of S_{16} . *CoRR*, abs/2609.17493, 2026. URL: <https://arxiv.org/abs/2609.17493>.