LETTER

Finding a Reconfiguration Sequence between Longest Increasing Subsequences

Yuuki AOIKE[†], **Masashi KIYOMI**[††a)], **Yasuaki KOBAYASHI**[†††b)], ***Nonmembers***, ***and* Yota OTACHI**[††††c)], ***Member***

SUMMARY In this note, we consider the problem of finding a step-by-step transformation between two longest increasing subsequences in a sequence, namely LONGEST INCREASING SUBSEQUENCE RECONFIGURATION. We give a polynomial-time algorithm for deciding whether there is a reconfiguration sequence between two longest increasing subsequences in a sequence. This implies that INDEPENDENT SET RECONFIGURATION and TOKEN SLIDING are polynomial-time solvable on permutation graphs, provided that the input two independent sets are largest among all independent sets in the input graph. We also consider a special case, where the underlying permutation graph of an input sequence is bipartite. In this case, we give a polynomial-time algorithm for finding a shortest reconfiguration sequence (if it exists).

key words: *combinatorial reconfiguration, longest increasing subsequence, permutation graph*

1. Introduction

For a nonnegative integer n, we define $[n] = \{1, 2, \ldots, n\}$. Let $A = (a_i)_{i=1,2,\ldots,n}$ be a sequence of distinct integers between 1 and n. We say that $I \subseteq [n]$ is *feasible* (for A) if $a_i < a_j$ for $i, j \in I$ with $i < j$. In other words, I is the set of indices of an increasing subsequence of A. A *maximum feasible set* (for A) is a feasible set I for A such that there is no feasible set (for A) with cardinality strictly larger than I. The problem of computing a maximum feasible set of a given sequence A, also known as LONGEST INCREASING SUBSEQUENCE, is a typical example that can be solved in polynomial time with dynamic programming [1].

In this note, we consider the reconfiguration-variant of LONGEST INCREASING SUBSEQUENCE, defined as follows. Given a sequence of n distinct integers A and (not necessarily maximum) two feasible sets I and J with $|I| = |J|$, the goal is to determine whether there is a sequence of feasible sets $I_0, I_1, \ldots, I_\ell$ such that $I_0 = I$, $I_\ell = J$, and for $1 \leq i \leq \ell$, I_i is obtained from I_{i-1} by simultaneously adding element $j \notin I_{i-1}$ and removing $k \in I_{i-1}$ (i.e., $I_i = (I_{i-1} \cup \{j\}) \setminus \{k\}$). We call this problem INCREASING SUBSEQUENCE RECONFIGURATION and such a sequence *a reconfiguration sequence between I and J*. If two input sets are maximum feasible sets for A, we particularly call the problem LONGEST INCREASING SUBSEQUENCE RECONFIGURATION. In this paper, we give a polynomial-time algorithm for LONGEST INCREASING SUBSEQUENCE RECONFIGURATION.

Theorem 1. LONGEST INCREASING SUBSEQUENCE RECONFIGURATION *can be solved in polynomial time.*

INCREASING SUBSEQUENCE RECONFIGURATION can be seen as a special case of a well-studied reconfiguration problem, called INDEPENDENT SET RECONFIGURATION. Given a graph $G = (V, E)$ and two independent sets I, J of G with $|I| = |J|$, INDEPENDENT SET RECONFIGURATION asks whether there is a sequence of independent sets $I_0, I_1, \ldots, I_\ell$ such that $I_0 = I$, $I_\ell = J$, and for $1 \leq i \leq \ell$, $I_i \setminus I_{i-1} = \{v\}$ and $I_{i-1} \setminus I_i = \{u\}$ for some $u, v \in V$. INCREASING SUBSEQUENCE RECONFIGURATION corresponds to INDEPENDENT SET RECONFIGURATION on permutations graphs: An undirected graph $G = (V, E)$ with $V = [n]$ is called a *permutation graph* if there is a permutation $\pi\colon [n] \to [n]$ such that for $1 \leq i < j \leq n$, $\pi(i) > \pi(j)$ if and only if $\{i, j\} \in E$. Observe that for $I \subseteq V$, I is an independent set of the permutation graph G if and only if I is a feasible set for $A = (\pi(i))_{i=1,2,\ldots,n}$. Thus, our problem, INCREASING SUBSEQUENCE RECONFIGURATION, is equivalent to INDEPENDENT SET RECONFIGURATION on permutation graphs. TOKEN SLIDING is a variant of INDEPENDENT SET RECONFIGURATION, where two vertices u, v in the above definition are required to be adjacent in G. It is easy to see that if I and J are maximum independent sets of G, these two problems are equivalent.

Corollary 1. INDEPENDENT SET RECONFIGURATION *and* TOKEN SLIDING *can be solved in polynomial time, provided that the input graph G is a permutation graph and two sets I and J are maximum independent sets of G.*

This resolves a special case of an open question posed by Briański et al. [2], where they ask for a polynomial-time algorithm for TOKEN SLIDING on permutation graphs.

The graph-theoretic perspective of LONGEST INCREASING SUBSEQUENCE RECONFIGURATION gives another interesting consequence of finding a *shortest* reconfiguration

Manuscript received October 11, 2023.
Manuscript revised November 27, 2023.
Manuscript publicized December 11, 2023.

[†]The author is with Yokohama City University, Yokohama-shi, 236–0027 Japan.
[††]The author is with Seikei University, Musashino-shi, 180–8633 Japan.
[†††]The author is with Hokkaido University, Sapporo-shi, 060–0814 Japan.
[††††]The author is with Nagoya University, Nagoya-shi, 464–8601 Japan.
a) E-mail: kiyomi@st.seikei.ac.jp
b) E-mail: koba@ist.hokudai.ac.jp
c) E-mail: otachi@nagoya-u.jp

DOI: 10.1587/transinf.2023EDL8067

sequence between maximum independent sets on bipartite permutation graphs. For any reconfiguration sequence $(I_0, I_1, \ldots, I_\ell)$, we have $\ell \geq |I_0 \setminus I_\ell|$, as we can remove at most one element from $I_0 \setminus I_\ell$ in a single step. For bipartite permutation graphs, we can always find a reconfiguration sequence between maximum independent sets I and J with length $\ell = |I \setminus J|$ if there is a reconfiguration sequence between them.

Theorem 2. *Let G be a bipartite permutation graph and let I and J be maximum independent sets of G. Suppose that there is a reconfiguration sequence between I and J. Then, there is a reconfiguration sequence of length $|I \setminus J|$ between I and J.*

The proof of Theorem 2 implies a polynomial-time algorithm for the "shortest-sequence variant" of Longest Increasing Subsequence Reconfiguration when the underlying permutation graph of an input sequence A is restricted to be bipartite.

Related work

Independent Set Reconfiguration and Token Sliding are both known to be PSPACE-complete [3]–[6] and studied on many graph classes. Independent Set Reconfiguration is solvable in polynomial time on even-hole free graphs [6] and cographs [7], [8], while it is NP-complete on bipartite graphs [9]. Token Sliding is solvable in polynomial time on cographs [6], bipartite permutation graphs [10], and interval graphs [2], [11], while it is PSPACE-complete on split graphs [3] and bipartite graphs [9]. The result of [10] does not yield Theorem 2 since their polynomial-time algorithm may provide a non-shortest reconfiguration sequence on bipartite permutation graphs. We particularly emphasize that both reconfiguration problems remain PSPACE-complete even if two input independent sets are maximum independent sets of the input graph [4].

As mentioned above, Independent Set Reconfiguration can be solved in polynomial time on the class of even-hole free graphs. In fact, for even-hole free graphs, Kamiński et al. [6] showed that every instance of Independent Set Reconfiguration is a yes-instance (assuming that two input independent sets have the same cardinality). This phenomenon does not hold on the class of permutation graphs: The instance consisting of $G := K_{2,2}$ with two color classes I and J is a no-instance, and G is indeed a permutation graph, corresponding to sequence $A = (7, 8, 5, 6)^\dagger$. Also both $I = \{1, 2\}$ and $J = \{3, 4\}$ are maximum independent sets of G. Thus, it is non-trivial to design a polynomial-time algorithm for Longest Increasing Subsequence Reconfiguration. Our polynomial-time algorithm exploits a structural property of the set of feasible sets for a given sequence A.

†For elements in A, we rather use integers more than n to distinguish from their indices in some concrete examples.

2. Algorithm

Let $A = (a_i)_{i=1,2,\ldots,n}$ be a sequence of n distinct integers between 1 and n. Let $V = [n]$ and let $P = (V, \preceq_A)$ be a partial order on V such that for $i, j \in V$,

$$i \preceq_A j \iff (i = j) \lor (i < j \land a_i < a_j).$$

Then, a subset of $[n]$ is feasible for A if and only if it is a chain of this partial order. Moreover, by Mirsky's theorem [12], the largest size of a chain of P is equal to the minimum size of an antichain partition of V, and such a partition can be computed in $O(n \log n)$ time by a standard dynamic programming algorithm for the longest increasing subsequence problem (see [13] for example).

To understand the structure of a minimum antichain partition, we use a specific construction, called *patience sorting* [14], which is briefly described as follows. For simplicity, we add 0 to A as $a_0 = 0$. We use $n+1$ piles $P_0, P_1, \ldots, P_n$ that are initially all empty and iteratively put an integer a_i in A on the top of one of the piles for $0 \leq i \leq n$ in this order. For each $0 \leq i \leq n$, we put a_i on the top of the "leftmost" pile P_j such that P_j is empty or the top of P_j is greater than a_i. Let us note that the top elements of all nonempty piles are always sorted in increasing order. Now, let $P_0, P_1, \ldots, P_n$ be the piles obtained by executing the above algorithm for A. Clearly, P_0 only contains a_0. For each pile P_i, observe that P_i is an antichain (with respect to $\preceq_A$): If a_i is placed below a_j in the pile, then $i < j$ and $a_i > a_j$. For each $1 \leq i \leq n$, when a_i is placed on the top of P_k for some $1 \leq k \leq n$, the top element a_j of P_{k-1} is smaller than a_i (i.e., $a_j < a_i$). In this case, we say that a_j *blocks* a_i and a_i *is blocked by* a_j. Let k be the largest index of a nonempty pile. By definition, for $1 \leq i \leq n$, each a_i has a unique element a_j that blocks a_i. Moreover, if a_i is blocked by a_j, we have $a_j \preceq_A a_i$. This implies that there is a chain (with respect to $\preceq_A$) of size $k+1$, which corresponds to a feasible set I for A. As each P_i is an antichain, this chain contains exactly one element of P_i for each $0 \leq i \leq k$. Thus, I is a maximum feasible set for A. The above construction of piles further implies the following observations.

Observation 1. *Let I be an arbitrary maximum feasible set for A.*

1. *If a_v is placed below a_u in a pile P_i, then we have $u > v$ and $a_u < a_v$.*
2. *Each pile P_i contains exactly one element a_u with $u \in I$.*
3. *Let $u, v \in I$ such that $a_u \in P_i$ and $a_v \in P_j$ for $0 \leq i < j \leq k$. Then, we have $u \preceq_A v$.*

Proof. The first statement follows from the construction of P_i. The second statement follows from the fact that P_i is an antichain with respect to $\preceq_A$. For the third statement, it suffices to show that $u < v$ (as the feasibility of I implies that $a_u < a_v$). Suppose for contradiction that $u > v$. When a_u is placed on the top of P_i, the top element on a pile P_j is strictly larger than a_u. This and the first statement together imply that $a_u < a_v$, contradicting the fact that I is feasible. □

Now, we turn to Longest Increasing Subsequence Reconfiguration. Let

$$\mathcal{I} = \{I \subseteq \{0\} \cup [n] : I \text{ is a maximum feasible set of } A\}$$

and let $P_0, P_1, \ldots, P_k$ be the nonempty piles that are obtained by applying the above algorithm to $A = (a_i)_{i=0,1,\ldots,n}$ with $a_0 = 0$. The following observation follows from (2) in Observation 1.

Observation 2. *Let $I, J \in \mathcal{I}$ such that $I \setminus J = \{u\}$ and $J \setminus I = \{v\}$. Then, $a_u, a_v \in P_j$ for some $0 \le j \le k$.*

Our algorithm for Longest Increasing Subsequence Reconfiguration is based on a certain equivalence relation on $\mathcal{I}$. For $I, J \in \mathcal{I}$, we denote by $I \lhd J$ if $I \setminus J = \{u\}$ and $J \setminus I = \{v\}$ such that a_u is placed (strictly) below a_v on pile P_i for some $1 \le i \le k$. We note that this $\lhd$ relation is not transitive: $I \lhd I'$ and $I' \lhd I''$ may not imply $I \lhd I''$. For $I \in \mathcal{I}$, a family of feasible sets $\mathcal{M}(I) \subseteq \mathcal{I}$ is defined inductively as follows: (1) $\mathcal{M}(I)$ contains I and (2) for every $J \in \mathcal{M}(I)$, $J' \lhd J$ implies $J' \in \mathcal{M}(I)$. In other words, $\mathcal{M}(I)$ is the lower set of I in the transitive closure of $\lhd$ in $\mathcal{I}$. By definition, for $I \in \mathcal{I}$, $\mathcal{M}(J) \subsetneq \mathcal{M}(I)$ if $J \in \mathcal{M}(I)$ with $J \neq I$. We say that $I \in \mathcal{I}$ is *$\lhd$-minimal* if there is no $J \in \mathcal{I}$ with $J \lhd I$.

Lemma 1. *Let $I, J, J' \in \mathcal{I}$ such that $J \lhd I$, $J' \lhd I$, and $J \neq J'$. Then, at least one of the following conditions is satisfied: $J' \lhd J$, $J \lhd J'$, or there is $J'' \in \mathcal{I}$ such that $J'' \lhd J$ and $J'' \lhd J'$.*

Proof. Let $I \setminus J = \{u\}$, $J \setminus I = \{v\}$, $I \setminus J' = \{u'\}$, and $J' \setminus I = \{v'\}$. If u and u' belong to the same pile P_i, by Observation 2, v and v' belong to the same pile P_i. This implies either $J' \lhd J$ or $J \lhd J'$. Suppose otherwise. By Observation 2, v and v' belong to distinct piles and hence $v \neq v'$. We claim that $(J \setminus \{u'\}) \cup \{v'\}$ is a maximum feasible set, which symmetrically implies that $(J' \setminus \{u\}) \cup \{v\}$ is a maximum feasible set as well. Suppose for contradiction that $(J \setminus \{u'\}) \cup \{v'\}$ is not a feasible set. Since $J \setminus \{u'\}$ and $J' = (I \setminus \{u'\}) \cup \{v'\}$ are feasible, v and v' are the unique incomparable pair with respect to $\le_A$ in $(J \setminus \{u'\}) \cup \{v'\}$. We assume that a_v and $a_{v'}$ are contained in piles P_i and P_j with $i < j$, respectively. As $v, u' \in J$, by Observation 1, we have $a_v < a_{u'}$. Moreover, as $a_{v'}$ is placed below $a_{u'}$ in P_j, we have $a_{u'} < a_{v'}$ (by (1) in Observation 1). These together imply that $a_v < a_{v'}$. As a_v is placed below a_u in P_i, we have $v < u$ (by (1) in Observation 1). Moreover, by (3) in Observation 1, $u \le_A v'$ as $u, v' \in J'$. Thus, we have $v < v'$, contradicting the assumption that v and v' are incomparable with respect to $\le_A$. □

Lemma 2. *For $I \in \mathcal{I}$, there is exactly one $\lhd$-minimal set in $\mathcal{M}(I)$.*

Proof. We prove the lemma by induction on $|\mathcal{M}(I)|$. If $|\mathcal{M}(I)| = 1$, then I itself is the unique $\lhd$-minimal set in $\mathcal{M}(I)$. Suppose that $\mathcal{M}(I)$ contains at least two sets. If there is exactly one $J \in \mathcal{M}(I)$ with $J \lhd I$, by the induction hypothesis, $\mathcal{M}(J) \subsetneq \mathcal{M}(I)$ has a unique $\lhd$-minimal set, which is also the unique $\lhd$-minimal set in $\mathcal{M}(I)$. Otherwise, there are two $J, J' \in \mathcal{M}(I)$ such that $J \lhd I$ and $J' \lhd I$. By Lemma 1, at least one of the following conditions are satisfied: $J' \lhd J$, $J \lhd J'$, or there is $J'' \in \mathcal{I}$ such that $J'' \lhd J$ and $J'' \lhd J'$. If $J' \lhd J$, then $\mathcal{M}(J') \subseteq \mathcal{M}(J) \subsetneq \mathcal{M}(I)$. By induction, both $\mathcal{M}(J)$ and $\mathcal{M}(J')$ have unique $\lhd$-minimal sets, and as $\mathcal{M}(J') \subseteq \mathcal{M}(J)$, these two sets are identical. The case where $J \lhd J'$ is symmetric. Hence, suppose that there is $J'' \in \mathcal{I}$ such that $J'' \lhd J$ and $J'' \lhd J'$. By induction, $\mathcal{M}(J)$, $\mathcal{M}(J')$, and $\mathcal{M}(J'')$ have unique $\lhd$-minimal sets. Similarly, as $\mathcal{M}(J'') \subseteq \mathcal{M}(J)$ and $\mathcal{M}(J'') \subseteq \mathcal{M}(J')$, these three $\lhd$-minimal sets are identical, which completes the proof. □

The proof of Lemma 2 immediately implies the following corollary.

Corollary 2. *For $I, J \in \mathcal{I}$ with $I \lhd J$, the $\lhd$-minimal sets of $\mathcal{M}(I)$ and $\mathcal{M}(J)$ are identical.*

We define an equivalence relation on $\mathcal{I}$ based on the $\lhd$-minimality. By Lemma 2, the $\lhd$-minimal set in $\mathcal{M}(I)$ is uniquely determined for $I \in \mathcal{I}$. We say that two maximum feasible sets I and J are *$\lhd$-equivalent* if the $\lhd$-minimal set in $\mathcal{M}(I)$ is equal to that in $\mathcal{M}(J)$. The key to our algorithm is the following lemma.

Lemma 3. *Let $I, J \in \mathcal{I}$. Then, there is a reconfiguration sequence between I and J if and only if I and J are $\lhd$-equivalent.*

Proof. Suppose that there is a reconfiguration sequence $(I_0, I_1, \ldots, I_\ell)$ between $I_0 = I$ and $I_\ell = J$. We prove that all maximum feasible sets I_i belong to the same $\lhd$-equivalence class. By definition, either $I_i \lhd I_{i+1}$ or $I_{i+1} \lhd I_i$, implying respectively that $\mathcal{M}(I_i) \subseteq \mathcal{M}(I_{i+1})$ or $\mathcal{M}(I_{i+1}) \subseteq \mathcal{M}(I_i)$. By Corollary 2, their $\lhd$-minimal sets are identical, which proves the forward direction.

Suppose that I and J are $\lhd$-equivalent. Then, there is $I' \in \mathcal{M}(I) \cap \mathcal{M}(J)$. This implies that there are reconfiguration sequences between I and I' and between J and I'. By concatenating these sequences, we have a reconfiguration sequence between I and J. □

Our algorithm is fairly straightforward. Given two maximum feasible sets I and J, we compute their $\lhd$-minimal sets I' and J', respectively. By Lemma 3, there is a reconfiguration sequence between I and J if and only if $I' = J'$. From a maximum feasible set I, we can compute a unique $\lhd$-minimal set in $\mathcal{M}(I)$ in polynomial time by a greedy algorithm. Hence, Theorem 1 follows.

3. Bipartite Case

Before proving Theorem 2, we would like to mention that bipartiteness in Theorem 2 is crucial, that is, Longest Increasing Subsequence Reconfiguration does not admit a reconfiguration sequence of length $|I \setminus J|$ in general. Let us consider an instance consisting of $A = (15, 11, 16, 13, 17, 12, 14)$[†], $I = \{1, 3, 5\}$, and $J = \{2, 6, 7\}$. This instance requires four steps to transform I into J: $I_0 = \{1, 3, 5\} = I$, $I_1 = \{2, 3, 5\}$, $I_2 = \{2, 4, 5\}$, $I_3 = \{2, 4, 7\}$, $I_4 = \{2, 6, 7\} = J$, while $|I \setminus J| = 3$.

Let $(A = (a_i)_{i=1,2,\ldots,n}, I, J)$ be an instance of Longest Increasing Subsequence Reconfiguration such that the underlying permutation graph G_A of A is bipartite. In the following, we may not distinguish the elements of A from their indices and then also refer to the elements of A as the vertices of G_A. Let $P_1, P_2, \ldots, P_k$ be the piles for A defined in the previous section. By (1) in Observation 2, every pair of indices of elements in a pile is incomparable with respect to $\preceq_A$. This implies that they are adjacent in the permutation graph G_A. Thus, each pile contains at most two elements as otherwise G_A contains a triangle. A pile P_t is called a *mixed pile* if it contains exactly two elements a_i and a_j with $i \in I$ and $j \in J$. Note that, for such a mixed pile P_t, both $j \notin I$ and $i \notin J$ hold. A pair of two mixed piles is called a *forbidden pair* if the four vertices corresponding to two mixed piles induce a cycle of length 4 in G_A. It is easy to observe that (A, I, J) is a no-instance if it has a forbidden pair.

A mixed pile P_i is called the *leftmost* mixed pile if no pile P_j with $j < i$ is mixed. The following lemma is a key to proving Theorem 2.

Lemma 4. *Suppose that (A, I, J) has no forbidden pairs. Let a_i, a_j be the elements in the leftmost mixed pile P_t with $i \in I$ and $j \in J$. Then, at least one of $(I \setminus \{i\}) \cup \{j\}$ or $(J \setminus \{j\}) \cup \{i\}$ is feasible.*

Proof. Suppose that both $I' = (I \setminus \{i\}) \cup \{j\}$ and $J' = (J \setminus \{j\}) \cup \{i\}$ are not feasible. As I' is not feasible, there is $i' \in I \setminus \{i\}$ that is adjacent to j in G_A. Let $P_{t'}$ be the pile containing $a_{i'}$. Since $j \in J$, pile $P_{t'}$ has an element $a_{j'}$ with $j' \in J$, which implies that $P_{t'}$ is a mixed pile with $t < t'$. Symmetrically, as J' is not feasible, there is a mixed pile $P_{t''}$ with $t < t''$ that has an element $a_{j''}$ with $j'' \in J \setminus \{j\}$ adjacent to i in G_A. If $t' = t''$, the pair P_t and P'_t forms a forbidden pair, contradicting the assumption. Assume, without loss of generality, that $t < t' < t''$. Since there are edges between j and i' and between i and j'', we have $a_j > a_{i'}$ and $a_i > a_{j''}$. As $j, j'' \in J$, we have $a_j < a_{j''}$. Thus, we have $a_{i'} < a_j < a_{j''} < a_i$, contradicting to the fact $a_i < a_{i'}$ as $i, i' \in I$. □

It would be worth mentioning that Lemma 4 is similar to Lemma 6 in [6], where they showed that if G is even-hole-free, the subgraph of G induced by $I \triangle J = (I \setminus J) \cup (J \setminus I)$ has no cycles and then there always exists a reconfiguration sequence between two independent sets I and J with the same cardinality. However, the subgraph of G_A induced by $I \triangle J$ may contain a cycle, even when it excludes forbidden pairs. See Fig. 1, for an illustration.

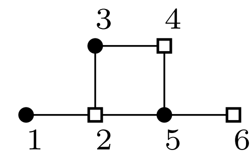

Fig. 1 The figure depicts the bipartite permutation graph G_A corresponding to sequence $A = (10, 7, 11, 8, 12, 9)$ with $I = \{1, 3, 5\}$ and $J = \{2, 4, 6\}$.

By Lemma 4, at least one of $(I \setminus \{i\}) \cup \{j\}$ or $(J \setminus \{j\}) \cup \{i\}$, say $I' = (I \setminus \{i\}) \cup \{j\}$, is feasible. This decreases the difference $|I' \setminus J|$ by 1 and does not create a new forbidden pair. Applying repeatedly this, Theorem 2 follows.

Acknowledgements

We appreciate anonymous reviewers for their careful reading of our manuscript and valuable comments. This work was partially supported by JSPS Kakenhi Grant Numbers JP20H00595, JP21K11752, JP22H00513, and JP23H03344.

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed., The MIT Press, 2022.

[2] M. Briański, S. Felsner, J. Hodor, and P. Micek, "Reconfiguring Independent Sets on Interval Graphs," 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021), ed. F. Bonchi and S.J. Puglisi, Leibniz International Proceedings in Informatics (LIPIcs), vol.202, Dagstuhl, Germany, pp.23:1–23:14, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[3] R. Belmonte, E.J. Kim, M. Lampis, V. Mitsou, Y. Otachi, and F. Sikora, "Token sliding on split graphs," Theory Comput. Syst., vol.65, no.4, pp.662–686, 2021.

[4] P. Bonsma and L. Cereceda, "Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances," Theor. Comput. Sci., vol.410, no.50, pp.5215–5226, 2009.

[5] R.A. Hearn and E.D. Demaine, "PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation," Theor. Comput. Sci., vol.343, no.1-2, pp.72–96, 2005.

[6] M. Kamiński, P. Medvedev, and M. Milanič, "Complexity of independent set reconfigurability problems," Theor. Comput. Sci., vol.439, pp.9–15, 2012.

[7] M. Bonamy and N. Bousquet, "Reconfiguring independent sets in cographs," CoRR, vol.abs/1406.1433, 2014.

[8] P. Bonsma, "Independent set reconfiguration in cographs and their generalizations," J. Graph Theory, vol.83, no.2, pp.164–195, 2016.

[9] D. Lokshtanov and A.E. Mouawad, "The complexity of independent set reconfiguration on bipartite graphs," ACM Trans. Algorithms, vol.15, no.1, pp.1–19, 2019.

[10] E. Fox-Epstein, D.A. Hoang, Y. Otachi, and R. Uehara, "Sliding token on bipartite permutation graphs," Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, Dec. 9-11, 2015, Proceedings, ed. K.M. Elbassioni and K. Makino, Lecture Notes in Computer Science, vol.9472, pp.237–247, Springer, 2015.

[11] M. Bonamy and N. Bousquet, "Token sliding on chordal graphs," Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23,

[†]Again, we use integers more than n for the elements in A to avoid confusion.

2017, Revised Selected Papers, ed. H.L. Bodlaender and G.J. Woeginger, Lecture Notes in Computer Science, vol.10520, pp.127–139, Springer, 2017.
[12] L. Mirsky, "A dual of Dilworth's decomposition theorem," The American Mathematical Monthly, vol.78, no.8, pp.876–877, 1971.
[13] M.L. Fredman, "On computing the length of longest increasing subsequences," Discret. Math., vol.11, no.1, pp.29–35, 1975.
[14] D. Aldous and P. Diaconis, "Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem," Bulletin of the American Mathematical Society, vol.36, pp.413–432, 1999.