Faster algorithms for the Permutation Pattern Matching Problem

(Szybsze algorytmy dla problemu wyszukiwania wzorca permutacyjnego)

Mateusz Rzepecki

Praca licencjacka

Promotor: dr Paweł Gawrychowski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

28 czerwca 2021

Abstract

Permutations and their properties are studied in many areas of science. For two permutations σ and π arises a very natural and simple to formulate question: does π occur in σ as a subpermutation and if so, then what is the number of such occurrences. It was shown by Bose, Buss and Lubiw [4] that the detection problem is NP-complete. This thesis shows three solutions to the counting problem. First works for $k = n^b$, such that $0 < b < 1$, any $c \leq \frac{1}{3}$ and any $\varepsilon > 0$ in time $\mathcal{O}\left(n^{k(1-b)\left(\frac{1}{3}+\frac{c}{3}+\varepsilon\right)}\right)$ and uses $\mathcal{O}\left(n^{k\left(1-b\right)\left(\frac{1-2c}{3}+\varepsilon\right)}\right)$ space. For $\frac{1}{4} < b$ it improves result by Berendsohn, Kozma and Marx [3], which works in time $\mathcal{O}(n^{\frac{1}{4}k+o(k)})$. Second algorithm works in time $\mathcal{O}(n^2\binom{\lceil\frac{n+k}{2}\rceil}{\lfloor\frac{n-k}{2}\rfloor})$ and uses $\mathcal{O}(n)$ space. Third algorithm works in time $\mathcal{O}(n \cdot \binom{\lfloor\frac{n}{2}\rfloor}{\lfloor\frac{k}{2}\rfloor})$ and uses $\mathcal{O}(n)$ space. Its time complexity can be also bounded by $\mathcal{O}(n \cdot 2^{\lfloor\frac{n}{2}\rfloor})$, therefore it improves the result by Berendsohn, Kozma and Marx [3], who designed an algorithm which works in time $\mathcal{O}(1.6181^n)$.

Permutacje i ich własności są obiektem badań w wielu dziedzinach nauki. Dla danych dwóch permutacji σ oraz π nasuwa się dość naturalne i proste do sformułowania pytanie: czy π występuje jako podpermutacja w σ i jeśli tak, to ile jest łącznie takich wystąpień. Bose, Buss i Lubiw [4] pokazali, że jest to problem NP-zupełny. Ta praca pokazuje trzy algorytmy rozwiązujące problem zliczania. Pierwszy z nich działa dla $k = n^b$, takiego że $0 < b < 1$, dowolnego $c \leq \frac{1}{3}$ oraz dowolnego $\varepsilon > 0$ w czasie $\mathcal{O}\left(n^{k(1-b)\left(\frac{1}{3}+\frac{c}{3}+\varepsilon\right)}\right)$ i używa $\mathcal{O}\left(n^{k\left(1-b\right)\left(\frac{1-2c}{3}+\varepsilon\right)}\right)$ pamięci. Dla $\frac{1}{4} < b$ poprawia on wynik Berendsohna, Kozmy i Marxa [3], którzy pokazali algorytm działający w czasie $\mathcal{O}(n^{\frac{1}{4}k+o(k)})$. Drugi algorytm działa w czasie $\mathcal{O}(n^2\binom{\lceil\frac{n+k}{2}\rceil}{\lfloor\frac{n-k}{2}\rfloor})$ i używa $\mathcal{O}(n)$ pamięci. Trzeci natomiast działa w czasie $\mathcal{O}(n \cdot \binom{\lfloor\frac{n}{2}\rfloor}{\lfloor\frac{k}{2}\rfloor})$ i używa $\mathcal{O}(n)$ pamięci. Jego czas działania można również oszacować przez $\mathcal{O}(n \cdot 2^{\lfloor\frac{n}{2}\rfloor})$, co poprawia wynik Berendsohna, Kozmy i Marxa [3], którzy pokazali algorytm działający w czasie $\mathcal{O}(1.6181^n)$.

Contents

Chapter 1

Introduction

Permutations are fundamental objects in mathematics and computer science. Permutation Pattern Matching Problem is to decide, for a given length-n permutation σ and length-k permutation π, if π occurs in σ as a subpermutation. The counting version of this problem asks about the number of different occurrences. For instance, for $\sigma = (3, 2, 5, 4, 1)$ and $\pi = (1, 3, 2)$ the answer to the decision problem is yes and answer to the counting problem is 2, since subpermutations of σ that have the same ordering as π are $(3, 5, 4)$, $(2, 5, 4)$.

Bose, Buss and Lubiw [4] showed that Permutation Pattern Matching Problem is NP-complete, thus we do not expect an efficient algorithm that solves this problem to exist. We can ask for an algorithm that works efficiently in special cases. Guillemot and Marx [9] showed that the detection problem can be solved in time $\mathcal{O}(n \cdot 2^{\mathcal{O}(k^2 \log k)})$, which works very well for small k. Later, Fox [8] cleansed the proof of Marcus and Tardos [11], what implied that the algorithm by Guillemot and Marx [9] works in time $\mathcal{O}(n \cdot 2^{\mathcal{O}(k^2)})$. There is also a very interesting reduction by Dudek and Gawrychowski [6] which states that counting cycles of length four in a sparse undirected graph is equivalent to counting the number of occurrences of length-4 permutations in a given permutation.

There is a trivial algorithm for the counting problem that works in time $\mathcal{O}(n^{k+1})$. The first algorithm that improved it was designed by Albert, Aldred, Atkinson and Holton [2] and works in time $\mathcal{O}(n^{\frac{2}{3}k+1})$. Their result was improved by Ahal and Rabinovich [1], who designed an algorithm that works in time $\mathcal{O}(n^{0.47k+o(k)})$. Another improvement was made by Berendsohn, Kozma and Marx [3], who showed an algorithm that works in time $\mathcal{O}(n^{k/4+o(k)})$. They also showed that the existence of an algorithm for the counting problem that works in time $\mathcal{O}(f(k) \cdot n^{o\left(\frac{k}{\log k}\right)})$ contradicts exponential time hypothesis [10].

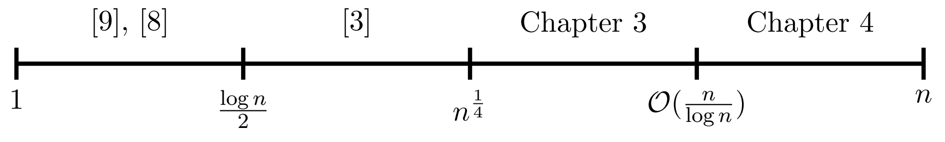
Berendsohn, Kozma and Marx [3] showed an algorithm for the counting problem that works in time $\mathcal{O}(1.6181^n)$ and uses polynomial space. Their algorithm guesses part of occurrence and then uses a polynomial-time algorithm to count the exact number of occurrences that match with the guessed values. This improved the result by Bruner and Lackner [5], who showed an algorithm working in time $\mathcal{O}(1.79^n)$.

We present an algorithm that works for $k = n^b$, such that $0 < b < 1$, any $c \leq \frac{1}{3}$ and any $\varepsilon > 0$ in time $\mathcal{O}\left(n^{k(1-b)\left(\frac{1}{3}+\frac{c}{3}+\varepsilon\right)}\right)$ and uses $\mathcal{O}\left(n^{k\left(1-b\right)\left(\frac{1-2c}{3}+\varepsilon\right)}\right)$ space. The idea is similar to the one by Berendsohn, Kozma and Marx [3]. Our algorithm splits $[k]$ into smaller segments, by guessing part of occurrence of π in σ, then it runs an algorithm that solves the counting version of binary Constraint Satisfaction Problem.

We introduce an algorithm that works in time $\mathcal{O}(n^2\binom{\lceil\frac{n+k}{2}\rceil}{\lfloor\frac{n-k}{2}\rfloor})$ and uses $\mathcal{O}(n)$ space. Our algorithm splits all occurrences of π in σ into groups based on part of positions in σ that do not match the pattern. Then, each group is solved separately with a polynomial-time algorithm which is an extension of the algorithm showed by Berendsohn, Kozma and Marx [3].

We also show an algorithm that works in time $\mathcal{O}(n \cdot \binom{\lfloor\frac{n}{2}\rfloor}{\lfloor\frac{k}{2}\rfloor})$ and uses $\mathcal{O}(n)$ space. The main improvement is achieved by generalizing the polynomial-time algorithm showed by Berendsohn, Kozma and Marx [3] even more. Our algorithm splits all possible occurrences of π in σ into groups based on approximate positions in σ of some elements in π. Then, each group is solved separately with the improved polynomial-time algorithm. We show that for specific properties of grouping occurrences, our split is almost optimal.

The diagram below presents the fastest algorithms for different values of k.



Chapter 2

Preliminaries

We denote $\{1, \ldots, n\}$ by $[n]$ and $\{n, \ldots, m\}$ by $[n, m]$. Permutation σ of length n is a bijection $\sigma : [n] \to [n]$. We call it also a length-n permutation. For a given $a \in [n]$ we denote a value of σ in a by $\sigma[a]$ and a value of σ^{-1} in a by $\sigma^{-1}[a]$. For a given set $\{a_1, \ldots, a_m\} = A \subseteq [n]$ we denote an array $\sigma[a_1], \ldots, \sigma[a_m]$ by $\sigma[A]$, where $a_i < a_{i+1}$ for $i \in [m-1]$. For a function f and a set B we denote a restriction of function f to the set B by $f|_B$. For functions f and g we write $\mathcal{O}(f) = \mathcal{O}(g)$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$.

Let σ be a length-n permutation and π be a length-k permutation. An instance of Permutation Pattern Matching Problem is a pair of permutations (σ, π). Solution to (σ, π) is an injection $f : [k] \to [n]$ such that:

1) $\forall_{i,j\in[k]} \pi[i] < \pi[j] \Leftrightarrow \sigma[f(i)] < \sigma[f(j)]$
2) $\forall_{i,j\in[k]} i < j \Leftrightarrow f(i) < f(j)$

Lemma 2.1. *Function $f : [k] \to [n]$ is a solution to (σ, π) if and only if:*

1) $\forall_{i\in[k-1]} \sigma[f(\pi^{-1}[i])] < \sigma[f(\pi^{-1}[i+1])]$, *we call them Y-axis constraints*
2) $\forall_{i\in[k-1]} f(i) < f(i+1)$, *we call them X-axis constraints*

Proof. It follows from the definition of a solution to (σ, π) and the fact that relation $<$ is transitive. □

Permutation Pattern Matching Problem	
Input:	length-n permutation σ and length-k permutation π
Output:	Does there exist an injection $f : [k] \to [n]$ such that:
	1) $\forall_{i,j\in[k]} \pi[i] < \pi[j] \Leftrightarrow \sigma[f(i)] < \sigma[f(j)]$
	2) $\forall_{i,j\in[k]} i < j \Leftrightarrow f(i) < f(j)$

Segment decomposition of an interval $[n]$ are segments $[l_1, r_1], \ldots, [l_m, r_m]$, such that $l_i \le r_i$ and $[l_i, r_i] \subseteq [m]$ for $i \in [m]$. Length of a segment decomposition is the

COUNTING PERMUTATION PATTERN MATCHING PROBLEM
Input: length-n permutation σ and length-k permutation π
Output: The number of injections $f : [k] \to [n]$ such that:
1) $\forall_{i,j\in[k]}\pi[i] < \pi[j] \Leftrightarrow \sigma[f(i)] < \sigma[f(j)]$
2) $\forall_{i,j\in[k]} i < j \Leftrightarrow f(i) < f(j)$

number of segments it consists of. We say that a segment decomposition is proper if $\bigcup_{i\in[m]}[l_i, r_i] = [n]$ and $r_i + 1 = l_{i+1}$ for $i \in [m-1]$. We say that a segment decomposition is 1-proper if $r_i \leq l_{i+1}$ for $i \in [m-1]$. See Figure 2.1 and Figure 2.2 as illustrations.

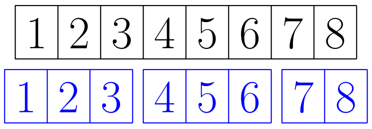


Figure 2.1: Black segments represent the interval [8] and blue segments represent a proper segment decomposition of [8], where $l_1 = 1$, $r_1 = 3$, $l_2 = 4$, $r_2 = 6$, $l_3 = 7$, $r_3 = 8$.

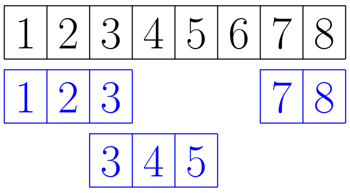


Figure 2.2: Black segments represent the interval [8] and blue segments represent a 1-proper segment decomposition of [8], where $l_1 = 1$, $r_1 = 3$, $l_2 = 3$, $r_2 = 5$, $l_3 = 7$, $r_3 = 8$.

Observation 2.2. *Proper segment decomposition is 1-proper.*

Observation 2.3. *In a proper segment decomposition* $l_1 = 1$ *and* $p_m = n$.

Segment decomposition with length m of (σ, π) are segment decomposition $[l_1, r_1], \ldots, [l_m, r_m]$ of $[n]$ with length m and a proper segment decomposition $[l'_1, r'_1], \ldots, [l'_m, r'_m]$ of $[k]$ with length m, such that $r_i - l_i \geq r'_i - l'_i$ for $i \in [m]$.

We define the width of a segment decomposition of (σ, π) by $\max_{i\in[m]}\{\min\{r'_i - l'_i + 1, (r_i - l_i + 1) - (r'_i - l'_i + 1)\}\}$. We call a segment decomposition of (σ, π) proper (1-proper) if its segment decomposition of $[n]$ is proper (1-proper).

Observation 2.4. *Each segment decomposition with width at most* 1 *satisfies* $r'_i - l'_i + 1 \leq 1 \lor (r_i - l_i + 1) - (r'_i - l'_i + 1) \leq 1$ *for* $i \in [m]$.

Definition 2.5. *Solution* f *to* (σ, π) *respects segment decomposition if* $f([l'_i, r'_i]) \subseteq [l_i, r_i]$ *for* $i \in [m]$. *See Figure 2.3 as an illustration.*

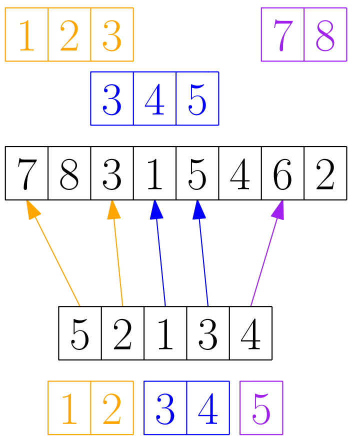


Figure 2.3: A solution f to (σ, π) that respects 1-proper segment decomposition of (σ, π). Black segments represent permutations $\sigma = (7, 8, 3, 1, 5, 4, 6, 2)$ and $\pi = (5, 2, 1, 3, 4)$. Orange, blue and purple segments represent the 1-proper segment decomposition of (σ, π), where $l_1 = 1$, $r_1 = 3$, $l_2 = 3$, $r_2 = 5$, $l_3 = 7$, $r_3 = 8$, $l'_1 = 1$, $r'_1 = 2$, $l'_2 = 3$, $r'_2 = 4$, $l'_3 = r'_3 = 5$. Arrows represent the solution f to (σ, π), where $f(1) = 1$, $f(2) = 3$, $f(3) = 1$, $f(4) = 5$ and $f(5) = 7$.

Definition 2.6 (Tree Decomposition)**.** *For a given graph $G = (V, E)$, tree decomposition of G is a tree $T = (V', E')$ and $f : V' \to \mathcal{P}(V)$ such that:*

1) $\forall_{e \in E} \exists_{v' \in V'} e_1, e_2 \in f(v')$, *where e_1, e_2 are the nodes connected by the edge e*
2) $\forall_{v \in V} \{v' \in V' : v \in f(v')\}$ *is a nonempty connected subtree of T*

The width of a tree decomposition is defined by $\max_{v' \in V'}\{|f(v')| - 1\}$. The treewidth of a graph G is defined as the minimum width of all of its tree decompositions.

Path decomposition is defined by analogy with tree decomposition, but we add one more requirement, namely T needs to be a path (a special type of a tree). We define the width of a path decomposition and the pathwidth of a graph by analogy with tree decomposition.

Lemma 2.7. *For any tree decomposition T, f of a graph G we can construct a tree decomposition T', f' of G with the same width, such that the number of edges in T' is less than the number of edges in G. We can construct it in polynomial time.*

Proof. Let T', f' be the value returned by Algorithm 1 called with parameters G, T and f. Clearly, T' and f' are a tree decomposition of G. There exists a function from edges in G onto nodes in T', since while loop in Algorithm 1 stopped. As a result, the number of edges in G is greater than the number of edges in T'. Running time of Algorithm 1 is polynomial in the size of the input, since an endpoint of each edge in T can be changed at most as many times as the number of nodes in T and each edge and node can be removed at most once. $\square$

Algorithm 1

```
Input:  graph $G$ and its tree decomposition $T$, $f$
Output: tree decomposition of $G$

begin
    Create a function $g$ from edges in $G$ to nodes in $T$, such that for any $e$
      both endpoints of $e$ belong to $f(g(e))$
    ▷ Possibly one node is assigned to more than one edge
    while exists a node in $T$ that is not in the image of $g$ do
        $u \leftarrow$ node in $T$ that is not in the image of $g$
        $v \leftarrow$ any neighbour of $u$ in $T$
        Remove edge between $u$ and $v$ from $T$
        foreach edge in $T$ with one endpoint in $u$ do
            Change endpoint of this edge from $u$ to $v$
        end
        Remove $u$ from $T$
    end
    $f \leftarrow f$ restricted to the nodes in $T$
    return $T$, $f$
end
```

An instance of binary Constraint Satisfaction Problem are variables $X_1, \dots, X_n$, domains of the variables $D_1, \dots, D_n$ and constraints $C_1, \dots, C_m$, where each C_i is represented by (i_1, i_2, R_i), where $i_1, i_2 \in [n]$ and $R_i \subseteq D_{i_1} \times D_{i_2}$. Solution to the instance of binary Constraint Satisfaction Problem is any array $(d_1, \dots, d_n)$ such that:

1) $\forall_{i \in [n]} d_i \in D_i$
2) $\forall_{i \in [m]} (d_{i_1}, d_{i_2}) \in R_i$

We define a constraint graph of the instance of binary Constraint Satisfaction Problem as a graph $G = (V, E)$, where nodes correspond to variables and for each constraint $C_i = (i_1, i_2, R_i)$ there is exactly one edge connecting nodes corresponding to the variables i_1 and i_2.

BINARY CONSTRAINT SATISFACTION PROBLEM
Input: variables $X_1, \dots, X_n$, domains of the variables $D_1, \dots, D_n$ and constraints $C_1, \dots, C_m$, where each C_i is represented by (i_1, i_2, R_i), where $i_1, i_2 \in [n]$ and $R_i \subseteq D_{i_1} \times D_{i_2}$.
Output: Does there exist an array $(d_1, \dots, d_n)$ such that:
1) $\forall_{i \in [n]} d_i \in D_i$
2) $\forall_{i \in [m]} (d_{i_1}, d_{i_2}) \in R_i$

Let $R_1, R_2 \subseteq [n] \times [n]$ be binary relations, such that:

1) $(a, b) \in R_1 \Leftrightarrow a < b$

Counting binary Constraint Satisfaction Problem	
Input:	variables $X_1, \ldots, X_n$, domains of the variables $D_1, \ldots, D_n$ and constraints $C_1, \ldots, C_m$, where each C_i is represented by (i_1, i_2, R_i), where $i_1, i_2 \in [n]$ and $R_i \subseteq D_{i_1} \times D_{i_2}$.
Output:	The number of arrays $(d_1, \ldots, d_n)$ such that: **1)** $\forall_{i \in [n]} d_i \in D_i$ **2)** $\forall_{i \in [m]} (d_{i_1}, d_{i_2}) \in R_i$

2) $(a, b) \in R_2 \Leftrightarrow \sigma[a] < \sigma[b]$

Let $X_1, \ldots, X_k$ be variables corresponding to the elements of π, $D_1 = D_2 = \ldots = D_k = [n]$, $C_i = (i, i+1, R_1)$, $C_{i+k-1} = (\pi^{-1}[i], \pi^{-1}[i+1], R_2)$ for $i \in [k-1]$.

Definition 2.8. *Variables $X_1, \ldots, X_k$, domains $D_1, \ldots, D_k$ and constraints $C_1, \ldots, C_{2k-2}$ are the instance of binary Constraint Satisfaction Problem corresponding to (σ, π). See Figure 2.4 for an illustration.*

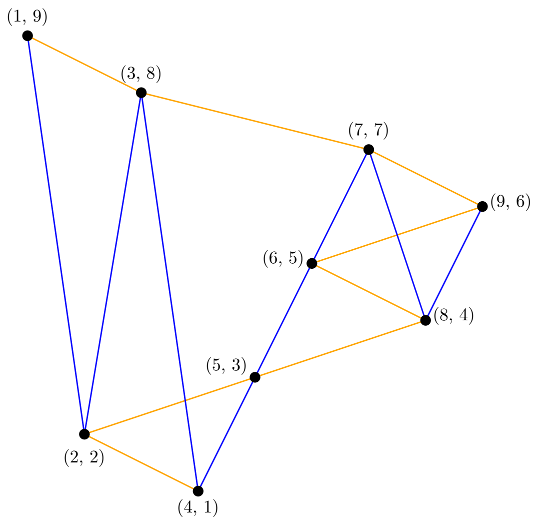


Figure 2.4: Constraint graph for $\pi = (9, 2, 8, 1, 3, 5, 7, 4, 6)$, where description of each node is $(i, \pi[i])$. Blue edges correspond to X-axis constraints and orange edges correspond to Y-axis constraints.

Lemma 2.9. *There exists a bijection between solutions to (σ, π) and solutions to the instance of binary Constraint Satisfaction Problem corresponding to it.*

Proof. Let A be a set of solutions to (σ, π) and B be a set of solutions to the instance of binary Constraint Satisfaction Problem corresponding to (σ, π).

For any $f \in A$ we know that $(f(1), f(2), \ldots, f(k)) \in B$, since

1) $\forall_{i \in [k]} f(i) \in [n] = D_i$,

2) $\forall_{i \in [k-1]} f(i) < f(i+1)$, thus $\forall_{i \in [k-1]} (f(i), f(i+1)) \in R_1$,

3) $\forall_{i\in[k-1]}\ \sigma[f(\pi^{-1}[i])] < \sigma[f(\pi^{-1}[i+1])]$, thus $\forall_{i\in[k-1]}(f(\pi^{-1}[i]), f(\pi^{-1}[i+1])) \in R_2$.

Thus, there exists a function $h_1 : A \to B$, such that $h_1(f) = (f(1), f(2), \ldots, f(k))$.

For any $(d_1, \ldots, d_k) \in B$ let $g : [k] \to [n]$ be defined by $g(i) = d_i$, then $g \in A$, since

1) $\forall_{i\in[k-1]}\ \sigma[g(\pi^{-1}[i])] < \sigma[g(\pi^{-1}[i+1])]$, since constraints $C_k, \ldots, C_{2k-2}$ are fulfilled,

2) $\forall_{i\in[k-1]} g(i) < g(i+1)$, since constraints $C_1, \ldots, C_{k-1}$ are fulfilled.

From Lemma 2.1 function g is a solution to (σ, π), thus there exists a function $h_2 : B \to A$, such that $h_2((d_1, \ldots, d_k)) = g$, where $g(i) = d_i$.

h_1 is a bijection, since $h_1 \circ h_2$ is an identity function. $\square$

As a result, instead of counting the number of solutions to the instance of Permutation Pattern Matching Problem, we can count the number of solutions to the instance of binary Constraint Satisfaction Problem corresponding to it.

Chapter 3

Constraint graph method

3.1 Binary Constraint Satisfaction Problem

Let $X_1, \ldots, X_a$, $D_1, \ldots, D_a$ and $C_1, \ldots, C_b$ be an instance of binary Constraint Satisfaction Problem. Let $G = (V, E)$ be its constraints graph. Let T be a tree decomposition of G with width t. Let **Root** be an arbitrary node in T and let T be rooted in **Root**. Let e be equal to the number of edges in T.

By nodes we mean nodes in T and by variables, we mean nodes in G and variables in the instance of binary Constraint Satisfaction Problem that correspond to those nodes. By $\mathbf{Var}(u)$ we denote a set of variables that are assigned to the node u. By $\mathbf{Subtree}(u)$ we denote nodes in the subtree of the node u (including u). By $\mathbf{Varsub}(u)$ we denote $\bigcup_{v \in \mathbf{Subtree}(u)} \mathbf{Var}(v)$. By $\mathbf{Children}(u)$ we denote all children of the node u in T. We call a function $f : A \to B$ a mapping if $A \subseteq [a]$ and $f(i) \in D_i$ for $i \in A$. By $\mathbf{Mapping}(A)$ we denote a set of mappings from a set A. By $\mathbf{Correct}(u, m)$ we denote a function that is equal to 1 if $m \in \mathbf{Mapping}(\mathbf{Var}(u))$ and assignment of values given by m fulfills all constraints in the node u and 0 otherwise. By $\mathbf{Con}(u)$ we denote the number of constraints in the node u. By $\mathbf{Deg}(u)$ we denote the number of children of the node u. By **Nodes** we denote the set of nodes in T.

Definition 3.1 (**DP**). *For $u \in$ **Nodes** and $m \in \mathbf{Mapping}(\mathbf{Var}(u))$ we define $\mathbf{DP}_{u,m}$ as the number of mappings $g \in \mathbf{Mapping}(\mathbf{Varsub}(u))$, such that all constraints in the subtree of u are fulfilled by the assignment of values given by g and $g|_{\mathbf{Var}(u)} = m$.*

Definition 3.2 (**Mutual**). *For $u, v \in$ **Nodes** and $m \in \mathbf{Mapping}(\mathbf{Var}(u) \cap \mathbf{Var}(v))$, where u is a parent of v we define $\mathbf{Mutual}_{u,v,m}$ as the number of mappings $g \in \mathbf{Mapping}(\mathbf{Varsub}(v))$, such that all constraints in the subtree of v are fulfilled by the assignment of values given by g and $g|_{\mathbf{Var}(u) \cap \mathbf{Var}(v)} = m$.*

Let A be a multiset $\{|D_i| : i \in [a]\}$, M_i be equal to the i-th biggest value in A,

Algorithm 2

```
Input:  node $u$ in $T$
Output: $\mathbf{DP}_{u,m}$ filled in with values, where $m \in \mathbf{Mapping}(\mathbf{Var}(u))$

begin
    foreach $v \in \mathbf{Children}(u)$ do
        Make a recursive call in $v$
        foreach $m \in \mathbf{Mapping}(\mathbf{Var}(v))$ do
            $m' \leftarrow m|_{\mathbf{Var}(u) \cap \mathbf{Var}(v)}$
            $\mathbf{Mutual}_{u,v,m'} \leftarrow \mathbf{DP}_{v,m} + \mathbf{Mutual}_{u,v,m'}$
        end
    end
    foreach $m \in \mathbf{Mapping}(\mathbf{Var}(u))$ do
        $\mathbf{DP}_{u,m} \leftarrow 1$
        foreach $v \in \mathbf{Children}(u)$ do
            $m_v \leftarrow m|_{Var(u) \cap Var(v)}$
            $\mathbf{DP}_{u,m} \leftarrow \mathbf{DP}_{u,m} \cdot \mathbf{Mutual}_{u,v,m_v}$
        end
        $\mathbf{DP}_{u,m} \leftarrow \mathbf{DP}_{u,m} \cdot \mathbf{Correct}(u, m)$
    end
end
```

$M = M_1 \cdot M_2 \cdot \ldots \cdot M_{t+1}$ and $\mathbf{Dom}_u = |\mathbf{Mapping}(\mathbf{Var}(u))|$.

Lemma 3.3. $\mathbf{Dom}_u \leq M$ *for* $u \in \mathbf{Nodes}$.

Proof. $\mathbf{Dom}_u \leq M_1 \cdot M_2 \cdot \ldots \cdot M_{t+1} = M$, since there are at most $t+1$ variables in each node. □

Lemma 3.4. *Algorithm 2 called parameter* **Root** *computes all values of* **DP** *in time* $\mathcal{O}(M \cdot (b + (t+1)e))$.

Proof. For any $u, v \in \mathbf{Nodes}$, where u is a parent of v the process of computing $\mathbf{Mutual}_{u,v,m}$ for all m takes $\mathcal{O}(\mathbf{Dom}_v \cdot |\mathbf{Var}(v) \cap \mathbf{Var}(u)|)$ time, since Algorithm 2 iterates over each $g \in \mathbf{Mapping}(\mathbf{Var}(v))$, restricts it to $\mathbf{Var}(v) \cap \mathbf{Var}(u)$ and then updates value of **Mutual**. As a result, computing **Mutual** in all recursive calls takes $\mathcal{O}(e \cdot \max_u\{\mathbf{Dom}_u\} \cdot (t+1))$ time, since T has e edges and $|\mathbf{Var}(v) \cap \mathbf{Var}(u)| \leq |\mathbf{Var}(u)| \leq t+1$.

The process of computing $\mathbf{DP}_{u,m}$ for any $u \in \mathbf{Nodes}$ and any $m \in \mathbf{Mapping}(\mathbf{Var}(u))$ takes $\mathcal{O}(\mathbf{Con}(u) + |\mathbf{Var}(u)| \cdot \mathbf{Deg}(u))$ time, since Algorithm 2 computes value of $\mathbf{Correct}(u, m)$ and restricts m to $\mathbf{Var}(u) \cap \mathbf{Var}(v)$ for each $v \in \mathbf{Children}(u)$. Thus, the process of computing all values of **DP** takes

$$
\begin{aligned}
&\mathcal{O}(\sum_{u \in \mathbf{Nodes}} \sum_{m \in \mathbf{Mapping}(\mathbf{Var}(u))} (\mathbf{Con}(u) + |\mathbf{Var}(u)| \cdot \mathbf{Deg}(u))) \\
=&\mathcal{O}(\sum_{u \in \mathbf{Nodes}} \mathbf{Dom}_u(\mathbf{Con}(u) + |\mathbf{Var}(u)| \cdot \mathbf{Deg}(u))) \\
=&\mathcal{O}(\max_u\{\mathbf{Dom}_u\} \sum_{u \in \mathbf{Nodes}} (\mathbf{Con}(u) + |\mathbf{Var}(u)| \cdot \mathbf{Deg}(u))) \\
=&\mathcal{O}(\max_u\{\mathbf{Dom}_u\} \cdot (b + \sum_{u \in \mathbf{Nodes}} |\mathbf{Var}(u)| \cdot \mathbf{Deg}(u))) \\
=&\mathcal{O}(\max_u\{\mathbf{Dom}_u\} \cdot (b + (t+1) \sum_{u \in \mathbf{Nodes}} \mathbf{Deg}(u))) \\
=&\mathcal{O}(\max_u\{\mathbf{Dom}_u\} \cdot (b + (t+1)e))
\end{aligned}
$$

After applying Lemma 3.3 we get the stated thesis.

□

Lemma 3.5. *Algorithm 2 called with parameter* **Root** *computes all values of* **DP** *using $\mathcal{O}(M \cdot (e+1))$ space.*

Proof. Algorithm 2 stores arrays **DP** and **Mutual**. Size of **DP** is $\mathcal{O}((e+1) \cdot \max_u\{\mathbf{Dom}_u\})$. Size of **Mutual** is equal to $\mathcal{O}(e \cdot \max_u\{\mathbf{Dom}_u\})$, since for each edge in T we store $\mathcal{O}(\max_u\{\mathbf{Dom}_u\})$ values. Thus, the final space consumption is $\mathcal{O}(\max_u\{\mathbf{Dom}_u\} \cdot (e+1))$. After applying Lemma 3.3 we get the stated thesis. □

Algorithm 3

Input: tree decomposition of the constraint graph of an instance of binary Constraint Satisfaction Problem

Output: the number of solutions to binary Constraint Satisfaction Problem

```
begin
    Call Algorithm 2 in Root
    result ← 0
    foreach $m \in \mathbf{Mapping}(\mathbf{Val}(\mathbf{Root}))$ do
        result ← result + $\mathbf{DP}_{\mathbf{Root},m}$
    end
    return result
end
```

Lemma 3.6. *Algorithm 3 computes the number of solutions to the instance of binary Constraint Satisfaction Problem in time $\mathcal{O}(M \cdot (b+(t+1)e))$ and uses $\mathcal{O}(M \cdot (e+1))$ space.*

Proof. From the definition of **DP**, Algorithm 3 computes the number of solutions to the instance of binary Constraint Satisfaction Problem.

The time complexity of Algorithm 3 can be bounded by $\mathcal{O}(t_1 + \mathbf{Dom}_{\mathbf{Root}})$, where t_1 is the time complexity of Algorithm 2 called with parameter **Root**. From Lemma 3.3 we know that Algorithm 3 works in time $\mathcal{O}(M \cdot (b + (t+1)e))$.

Algorithm 3 uses the same amount of space as Algorithm 2 called with parameter **Root**, thus from Lemma 3.5 we obtain the stated space bound. □

3.2 Permutation Pattern Matching Problem

Proposition 3.7 ([7]). *For any $\varepsilon > 0$, there exists an integer n_ε such that for every connected graph G with $n > n_\varepsilon$ nodes and $m = \beta \cdot n$ edges, where $\frac{3}{2} \leq \beta \leq 2$, the treewidth of G is at most $\frac{m-n}{3} + \varepsilon \cdot n$. Moreover, a tree decomposition of the corresponding width can be constructed in polynomial time.*

Lemma 3.8. *The number of edges in the tree decomposition from Proposition 3.7 can be bounded by the number of edges in the original graph.*

Proof. This follows from Lemma 2.7 applied to the tree decomposition in Proposition 3.7. □

Let W be a polynomial such that the tree decomposition from Proposition 3.7 and Lemma 3.8 can be created in time $\mathcal{O}(W(n))$ using $\mathcal{O}(W(n))$ space, where n is the number of nodes in the graph.

Let σ be a length-n permutation and π be a length-k permutation. To avoid clutter, we assume that ck is an integer.

Lemma 3.9. *Algorithm 4 computes the number of solutions f to (σ, π), such that values $f(a_1), \ldots, f(a_{ck})$ are the same as the values of variables $X_{a_1}, \ldots, X_{a_{ck}}$ from the input.*

Proof. This follows from Corollary 2.9 and the fact that modifications to G and the instance of binary Constraint Satisfaction Problem corresponding to (σ, π) made by Algorithm 4 do not change the number of solutions to it. □

Lemma 3.10. *For any $\varepsilon > 0$ there exists n_ε such that if $k > n_\varepsilon$ then tree decomposition of a modified constraint graph in Algorithm 4 has treewidth at most $\frac{k-2ck}{3} + \varepsilon k$ and this tree decomposition can be constructed in $\mathcal{O}(W(k))$ time.*

Proof. Let X and Y be sets of the edges removed in Algorithm 4 that correspond to X-axis and Y-axis constraints, respectively. Let Z be a set of nodes removed in Algorithm 4. See Figure 3.1 as an illustration. Let G be the constraint graph

Algorithm 4

```
Input:  $\sigma$, $\pi$, constant $c \leq \frac{1}{3}$, integers $a_1, \dots, a_{ck}$ evenly distributed in the
        interval $[k]$ and values of variables $X_{a_1}, \dots, X_{a_{ck}}$ in the instance
        of the Constraint Satisfaction Problem corresponding to $(\sigma, \pi)$
Output: the number of solutions $f$ to $(\sigma, \pi)$, such that values
        $f(a_1), \dots, f(a_{ck})$ are the same as the values of variables
        $X_{a_1}, \dots, X_{a_{ck}}$ from the input

begin
    Create a constraint graph $G$ corresponding to $(\sigma, \pi)$
    Remove all constraints that contain at least one of the variables
      $X_{a_1}, \dots, X_{a_{ck}}$ from $G$
    Remove all nodes corresponding to the variables $X_{a_1}, \dots, X_{a_{ck}}$ from $G$
    Update domains of variables in $G$
    $\triangleright$ We call $G$ a modified constraint graph
    $T, f \leftarrow$ tree decomposition of $G$ stated in Proposition 3.7 and
      Lemma 3.8
    return value returned by Algorithm 3 called with parameter $T$ and $f$
end
```

from Algorithm 4 before removal of the nodes and the edges, G' be a graph G after removing the edges from set X, G'' be a graph G' after removing the edges from set Y and G''' be a graph G'' after removing the nodes from set Z.

Let T' be a spanning tree of G' that consists of the edges that correspond to Y-axis constraints. Let T'' be a subgraph of G'' obtained by removing the edges from set Y from T'. Let T''' be a subgraph of G''' obtained by removing the nodes from set Z from T''.

We observe that:

1) T' is a connected graph.
2) T'' consists of at most $|Y| + 1$ connected components.
3) there are ck connected components in T'' that are single nodes corresponding to variables $X_{a_1}, \dots, X_{a_{ck}}$.
4) $2ck - 2 \leq |X| \leq 2ck$, since $c \leq \frac{1}{3}$ and $a_1, \dots, a_{ck}$ are evenly distributed in the interval $[k]$.

From 1) we get that G' is a connected graph. From 3) and 4) we get that T''' consists of at most $|Y| + 1 - ck$ connected components, thus we can add at most $|Y| - ck$ edges to G''' to make it connected, since T''' is a subgraph of G''' with the same set of nodes.

Let H be a graph G''' with additional $|Y| - ck + 2 + (|X| - 2ck)$ edges, such that H is connected. H has exactly $k - ck$ nodes and $2k - 2 - |X| - |Y| + (|Y| - ck + 2 + (|X| - 2ck)) = 2k - 2 - 2ck - ck + 2 = 2k - 3ck$ edges. We can apply Proposition 3.7

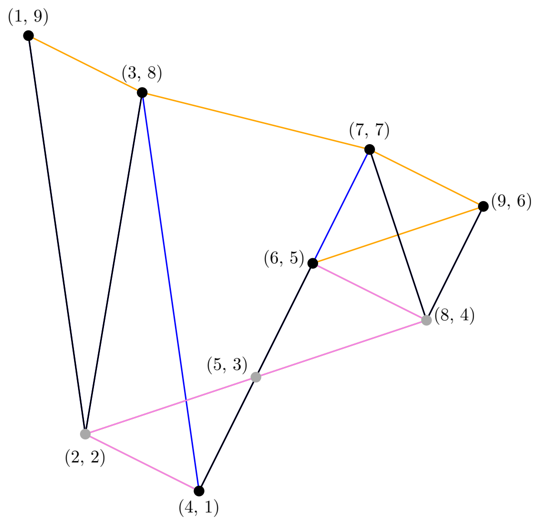


Figure 3.1: Constraint graph edited by Algorithm 4 for $\pi = (9, 2, 8, 1, 3, 5, 7, 4, 6)$ and $c = \frac{1}{3}$, where description of each node is $(i, \pi[i])$. Black edges correspond to removed X-axis constraints, pink edges correspond to removed Y-axis edges and gray nodes correspond to removed nodes. Blue edges correspond to not removed X-axis constraints and orange edges correspond to not removed Y-axis constraints.

and Lemma 3.8 to this graph, since $\frac{3}{2}(k - ck) \leq 2k - 3ck \leq 2(k - ck)$. We get that for any $\varepsilon > 0$ there exists n_ε such that if $k - ck > n_\varepsilon$ then the treewidth of H is at most $\frac{(2k-3ck)-(k-ck)}{3} + \varepsilon(k - ck) = \frac{k-2ck}{3} + \varepsilon(k - ck) \leq \frac{k-2ck}{3} + \varepsilon k$ and a tree decomposition of the corresponding width can be constructed in polynomial time. Constraint $k - ck > n_\varepsilon$ is implied by $k > 2n_\varepsilon$, since $c \leq \frac{1}{3}$. From the fact that G''' is a subgraph of H and they have the same set of nodes, we get the stated thesis. □

Lemma 3.11. *Algorithm 5 computes the number of solutions to* (σ, π).

Proof. This follows from Lemma 3.9 and the fact that for each solution to (σ, π) there exists exactly one possible assignment of values to the variables $X_{a_1}, \dots, X_{a_{ck}}$ that matches with this solution. □

Let m be an increasing function $m : \{a_1, \dots, a_{ck}\} \to [n]$ read by Algorithm 4. Let $\mathbf{D}_i$ be a domain of X_i in the modified constraint graph from Algorithm 4 constructed using $m(a_i)$ as a value of X_{a_i} for $i \in [ck]$. Let $\mathbf{Domain}_i = \bigcup_{j \in [a_{i-1}+1, a_i - 1]} \mathbf{D}_j$, where $a_0 = 0$ and $a_{ck+1} = k + 1$.

Lemma 3.12. $\mathbf{Domain}_i \subseteq \sigma[[m(a_{i-1}) + 1, m(a_i) - 1]]$ *for* $i \in [ck + 1]$.

Proof. Removed X-axis constraints imply that $\mathbf{D}_j \subseteq [m(a_{i-1}) + 1, m(a_i) - 1]$ for $j \in [a_{i-1} + 1, a_i - 1]$. From the definition of $\mathbf{Domain}_i$ we get the stated thesis. □

Lemma 3.13. $\mathbf{Domain}_i \cap \mathbf{Domain}_j = \emptyset$ *for* $i, j \in [ck + 1]$.

Algorithm 5

```
Input:  $\sigma$, $\pi$, constant $c \leq \frac{1}{3}$
Output: the number of solutions to $(\sigma, \pi)$

begin
    Set values of integers $a_1, \dots, a_{ck}$ evenly distributed in the interval $[k]$
    $G \leftarrow$ the instance of binary Constraint Satisfaction Problem
      corresponding to $(\sigma, \pi)$
    result $\leftarrow 0$
    foreach $m$ in increasing assignments of values to variables
     $X_{a_1}, \dots, X_{a_{ck}}$ in $G$ do
        result $\leftarrow$ result + value returned by Algorithm 4 called with
          parameters $\sigma$, $\pi$, $c$ and $m$
    end
    return result
end
```

Proof. Without loss of generality we assume that $i < j$. From Lemma 3.12 we know that $\mathbf{Domain}_i \subseteq \sigma[[m(a_{i-1})+1, m(a_i)-1]]$ and $\mathbf{Domain}_j \subseteq \sigma[[m(a_{j-1})+1, m(a_j)-1]]$. From the fact that $m(a_i) - 1 < m(a_{j-1}) + 1$ we get the stated thesis. □

Lemma 3.14. $\sum_{i=1}^{i=ck+1} |\mathbf{Domain}_i| \leq n$

Proof. From Lemma 3.13 we know that the sum $\bigcup_i \mathbf{Domain}_i \subseteq [n]$ consists of disjoint sets. This concludes the proof. □

Lemma 3.15. *For any integer t and a sequence of pairwise distinct integers $z_1, \dots, z_t$ from set $[k] \setminus \{a_1, \dots, a_{ck}\}$ the following inequality holds $\prod_{i=1}^{i=t} |\mathbf{D}_{z_i}| \leq \left(\frac{n}{ct}\right)^t$.*

Proof. Let $d = \frac{1}{c} = \frac{k-ck}{ck} + 1 > \lceil \frac{k-ck}{ck+1} \rceil$. For each $z_i \in [k] \setminus \{a_1, \dots, a_{ck}\}$ there exists exactly one y_i such that $\mathbf{D}_{z_i} \subseteq \mathbf{Domain}_{y_i}$, additionally $|\mathbf{D}_{z_i}| \leq |\mathbf{Domain}_{y_i}|$. For each i there are at most d values $j \in [t]$ such that $\mathbf{D}_j \subseteq \mathbf{Domain}_i$, thus $\sum_{i=1}^{i=t} |\mathbf{D}_{z_i}| \leq d \cdot \sum_{i=1}^{i=ck+1} |\mathbf{Domain}_i| \leq dn$. The last inequality follows from Lemma 3.14. As a result

$$\prod_{i=1}^{i=t} |\mathbf{D}_{z_i}| \leq \left(\frac{\sum_{i=1}^{i=t} |\mathbf{D}_{z_i}|}{t}\right)^t \leq \left(\frac{dn}{t}\right)^t = \left(\frac{n}{ct}\right)^t,$$

where the first inequality follows from AM-GM inequality. □

Lemma 3.16. *Algorithm 4 works in time $\mathcal{O}\Big(W(k) + k^2 \cdot \left(\frac{n}{c(t'+1)}\right)^{t'+1}\Big)$ and uses $\mathcal{O}\Big(W(k) + k \cdot \left(\frac{n}{c(t'+1)}\right)^{t'+1}\Big)$ space, for any $c \leq \frac{1}{3}$ and $\varepsilon > 0$, where $t' = \frac{k-2ck}{3} + \varepsilon k$.*

Proof. From Lemma 3.10 we know that for any $\varepsilon > 0$ the tree decomposition of the modified constraint graph in Algorithm 4 has treeewidth at most $t' = \frac{k-2ck}{3} + \varepsilon k$ or is less than a constant dependent only on ε. Lemma 3.15 applied for $t'+1$ combined with Lemma 3.6 yield a bound $\mathcal{O}\Big(\big(\frac{n}{c(t'+1)}\big)^{(t'+1)} \cdot (b + (t'+1)e)\Big)$ on the running time and $\mathcal{O}\Big(\big(\frac{n}{c(t'+1)}\big)^{(t'+1)} \cdot (e+1)\Big)$ on the space consumption of Algorithm 3 called in Algorithm 4, where b is the number of constraints and e is the number of edges in the tree decomposition in Algorithm 4. We observe that $b \le 2k$, $t'+1 \le 2k$ and $e \le b-1 \le 2k$, where the last two inequalities follow from Lemma 3.8). Time and space required to construct the tree decomposition of the modified constraint graph in Algorithm 4 is $\mathcal{O}(W(k))$, thus Algorithm 4 works in time $\mathcal{O}\Big(W(k) + \big(\frac{n}{c(t'+1)}\big)^{(t'+1)} \cdot (2k + 2k\cdot 2k)\Big)$ and uses $\mathcal{O}\Big(W(k) + \big(\frac{n}{c(t'+1)}\big)^{(t'+1)} \cdot (2k+1)\Big)$ space. □

Lemma 3.17. *Algorithm 5 works in time* $\mathcal{O}\Big(\binom{n}{ck} \cdot \Big(W(k) + k^2 \cdot \big(\frac{n}{c(t'+1)}\big)^{t'+1}\Big)\Big)$ *and uses* $\mathcal{O}\Big(W(k) + k \cdot \big(\frac{n}{c(t'+1)}\big)^{t'+1}\Big)$ *space, for any* $c \le \frac{1}{3}$ *and* $\varepsilon > 0$, *where* $t' = \frac{k-2ck}{3} + \varepsilon k$.

Proof. There are at most $\binom{n}{ck}$ possible increasing assignments of values to the variables $X_{a_1}, \dots, X_{a_{ck}}$ and we can iterate over all of them in time $\mathcal{O}(\binom{n}{ck})$ and $\mathcal{O}(n)$ space. For each increasing assignment Algorithm 5 calls Algorithm 4, thus from Lemma 3.16 we get the stated bounds. □

Fact 3.18. $\binom{n}{k} \le (\frac{e\cdot n}{k})^k$

Theorem 3.19. *For* $k = n^b$, *such that* $0 < b < 1$, *any* $c \le \frac{1}{3}$ *and any* $\varepsilon > 0$ *Counting Permutation Pattern Matching Problem can be solved in time* $\mathcal{O}\Big(n^{k(1-b)\left(\frac{1}{3}+\frac{c}{3}+\varepsilon\right)}\Big)$ *and* $\mathcal{O}\Big(n^{k\left(1-b\right)\left(\frac{1-2c}{3}+\varepsilon\right)}\Big)$ *space.*

Proof. From Lemma 3.11 we know that Algorithm 5 computes the number of solutions to (σ, π). From Lemma 3.17 we know that Algorithm 5 works in time $\mathcal{O}\Big(\binom{n}{ck} \cdot \Big(W(k) + k^2 \cdot \big(\frac{n}{c(t'+1)}\big)^{t'+1}\Big)\Big)$ and $\mathcal{O}\Big(W(k) + k \cdot \big(\frac{n}{c(t'+1)}\big)^{t'+1}\Big)$ space, for any $c \le \frac{1}{3}$ and $\varepsilon > 0$, where $t' = \frac{k-2ck}{3} + \varepsilon k$. We bound the time complexity using observations:

$$
\begin{aligned}
&\mathcal{O}\Big(\binom{n}{ck} \cdot \Big(W(k) + k^2 \cdot \Big(\frac{n}{c(t'+1)}\Big)^{t'+1}\Big)\Big) \\
=&\mathcal{O}\Big(\Big(\frac{en}{ck}\Big)^{ck} \cdot \Big(W(k) + k^2 \cdot \Big(\frac{n}{c(t'+1)}\Big)^{t'+1}\Big)\Big) && \text{follows from Fact 3.18} \\
\overset{(1)}{=}&\mathcal{O}\Big(\Big(\frac{en}{ck}\Big)^{ck} \cdot k^2 \cdot \Big(\frac{n}{c(t'+1)}\Big)^{t'+1}\Big) \\
\overset{(2)}{=}&\mathcal{O}\Big(\Big(\frac{en}{ck}\Big)^{ck} \cdot k^2 \cdot \Big(\frac{n}{c} \cdot \frac{3}{k} \cdot \frac{1}{1-2c}\Big)^{t'+1}\Big) \\
=&\mathcal{O}\Big(f(c)^k \cdot k^2 \cdot \Big(\frac{n}{k}\Big)^{t'+ck+1}\Big) && \text{follows from } t'+1 \le k
\end{aligned}
$$

$$
\begin{aligned}
&= \mathcal{O}\Big(f(c)^k \cdot n^{2b} \cdot n^{\left(1-b\right)\left(\frac{k+ck+3}{3}+\varepsilon k\right)}\Big) && \text{follows from } k = n^b \\
&\overset{(3)}{=} \mathcal{O}\Big(n^{\left(1-b\right)\left(\frac{k+ck}{3}+\varepsilon k\right)}\Big),
\end{aligned}
$$

where $f(c)$ is a function dependent on c. Explanations to some steps in the above calculations:

(1) follows from $\mathcal{O}(W(k)) = \mathcal{O}\Big(k^2 \cdot \left(\frac{n}{c(t'+1)}\right)^{t'+1}\Big)$, which is true, since $\frac{n}{c(t'+1)} > 1$ and $t'+1 \geq \frac{k}{9}$.

(2) follows from inequality $\frac{1}{t'+1} = \frac{1}{\frac{k-2ck+3}{3}+\varepsilon k} = \frac{3}{k} \cdot \frac{1}{1-2c+\frac{3}{k}+3\varepsilon} \leq \frac{3}{k} \cdot \frac{1}{1-2c}$.

(3) follows from ε being any constant greater than zero and $\mathcal{O}(f(c)^k \cdot n^{2b+(1-b)}) = \mathcal{O}(n^{k\delta})$ for any $\delta > 0$.

We bound the space consumption using observations:

$$
\begin{aligned}
&\mathcal{O}\Big(W(k) + k^2 \cdot \Big(\frac{n}{c(t'+1)}\Big)^{t'+1}\Big) \\
&\overset{(1)}{=} \mathcal{O}\Big(k^2 \cdot \Big(\frac{n}{c(t'+1)}\Big)^{t'+1}\Big) \\
&\overset{(2)}{=} \mathcal{O}\Big(k^2 \cdot \Big(\frac{n}{c} \cdot \frac{3}{k} \cdot \frac{1}{1-2c}\Big)^{t'+1}\Big) \\
&= \mathcal{O}\Big(g(c)^k \cdot k^2 \cdot \Big(\frac{n}{k}\Big)^{t'+1}\Big) && \text{follows from } t'+1 \leq k \\
&= \mathcal{O}\Big(g(c)^k \cdot n^{2b} \cdot n^{\left(1-b\right)\left(\frac{k-2ck+3}{3}+\varepsilon k\right)}\Big) && \text{follows from } k = n^b \\
&\overset{(3)}{=} \mathcal{O}\Big(n^{\left(1-b\right)\left(\frac{k-2ck}{3}+\varepsilon k\right)}\Big),
\end{aligned}
$$

where $g(c)$ is a function dependent on c.

Explanations to some steps in the above calculations:

(1) follows from $\mathcal{O}(W(k)) = \mathcal{O}\Big(k^2 \cdot \left(\frac{n}{c(t'+1)}\right)^{t'+1}\Big)$, which is true, since $\frac{n}{c(t'+1)} > 1$ and $t'+1 \geq \frac{k}{9}$.

(2) follows from inequality $\frac{1}{t'+1} = \frac{1}{\frac{k-2ck+3}{3}+\varepsilon k} = \frac{3}{k} \cdot \frac{1}{1-2c+\frac{3}{k}+3\varepsilon} \leq \frac{3}{k} \cdot \frac{1}{1-2c}$.

(3) follows from ε being any constant greater than zero and $\mathcal{O}(g(c)^k \cdot n^{2b+(1-b)}) = \mathcal{O}(n^{k\delta})$ for any $\delta > 0$.

This concludes the proof. $\square$

Chapter 4

Segment decompositions

4.1 Special case

Let σ be a length-n permutation and π be a length-k permutation, such that $k = n-1$. We assume there exists at least one solution to (σ, π). Each solution f to (σ, π) can be interpreted as a single number $c \in [n] \setminus f([k])$, since $|[n] \setminus f([k])| = 1$. Let a be the smallest and b be the biggest number that corresponds to a solution to (σ, π). See Figure 4.1 as an illustration.

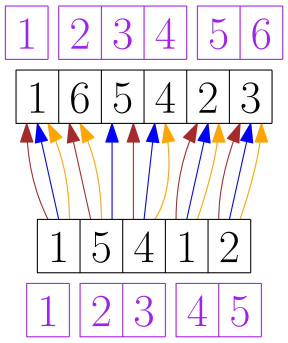


Figure 4.1: Situation from Section 4.1 for permutations $\sigma = (1,6,5,4,2,3)$ and $\pi = (1,5,4,1,2)$. Black segments represent permutations σ and π. Blue arrows represent a solution to (σ, π) that correspond to 2, orange to 3 and brown to 4. Purple blocks represent a proper segment decomposition of (σ, π) such that each solution to (σ, π) respects it. This follows from Lemma 4.2 and Lemma 4.3 and the fact that $a = 2$ and $b = 4$.

Lemma 4.1. *$\sigma[[a, b]]$ and $\pi[[a, b-1]]$ are monotonic.*

Proof. If $b < a + 2$ then the thesis is true, thus we assume that $a + 2 \leq b$. Let f be a solution to (σ, π) that corresponds to a and g be a solution to (σ, π) that corresponds to b. For $i \in [a, b-1]$ we know that $f(i) = i + 1$ and $g(i) = i$. As a result $\pi[i] < \pi[i+1] \Leftrightarrow \sigma[i+1] < \sigma[i+2]$ and $\pi[i] < \pi[i+1] \Leftrightarrow \sigma[i] < \sigma[i+1]$ for

$i \in [a, b-2]$, thus $\sigma[i] < \sigma[i+1] \Leftrightarrow \sigma[i+1] < \sigma[i+2]$ for $i \in [a, b-2]$. □

Lemma 4.2. *For any $i \in [a-1]$ and any solution f to (σ, π) we know that $f(i) = i$.*

Proof. If $f(i) \neq i$ for any $i \in [a-1]$, then $f(i) = i+1$, thus there exists $j \leq i < a$ such that $j \notin f([k])$. This contradicts with the definition of a. □

Lemma 4.3. *For any $i \in [b, k]$ and any solution f to (σ, π) we know that $f(i) = i+1$.*

Proof. If $f(i) \neq i+1$ for any $i \in [b, k]$, then $f(i) = i$, thus there exists $j \notin f([k])$, such that $b \leq i < j$. This contradicts with the definition of b. □

Lemma 4.4. *Any function f such that $f(i) = i$ for $i \in [a-1]$, $f(i) = i+1$ for $i \in [b, k]$, $f(i) \in [a, b]$ for $i \in [a, b-1]$ and f fulfills all Y-axis constraints is a solution to (σ, π).*

Proof. From Lemma 4.1 and the fact that f fulfills Y-axis constraints we know that X-axis constraints concerning interval $[a, b-1]$ are fulfilled. All other X-axis constraints are fulfilled as well, thus f is a solution to (σ, π). □

4.2 Greedy algorithm

Let σ be a length-n permutation and π be a length-k permutation. Let $[l_1, r_1], \ldots, [l_m, r_m]$ and $[l'_1, r'_1], \ldots, [l'_m, r'_m]$ be a 1-proper segment decomposition of (σ, π) with width at most 1. We call it A.

Definition 4.5. *For $i \in [k]$ and $j \in [n]$ we define $\mathbf{DP}_{i,j}$ as the number of functions $f : \pi^{-1}[[i]] \to [n]$, such that all Y-axis constraints concerning $\pi^{-1}[[i]]$ are fulfilled, $f(\pi^{-1}[i]) = j$ and f respects A. We denote an array that consists of $\mathbf{DP}_{i,1}, \ldots, \mathbf{DP}_{i,n}$ by $\mathbf{DP}_i$.*

Definition 4.6. *For $i \in [k]$ and $j \in [n]$ we define $\mathbf{Pref}_{i,j} = \sum_{m=1}^{m=j} \mathbf{DP}_{i,m}$. We denote an array that consists of $\mathbf{Pref}_{i,1}, \ldots, \mathbf{Pref}_{i,n}$ by $\mathbf{Pref}_i$.*

We are now ready to present Algorithm 6 that counts the number of solutions to (σ, π), see the pseudocode below.

Lemma 4.7. *B in Algorithm 6 is a 1-proper segment decomposition of (σ, π) with width at most 1. Moreover, a solution to (σ, π) respects A if and only if it respects B.*

Proof. B is a 1-proper segment decomposition of (σ, π), since A is a 1-proper segment decomposition and segments added to B do not intersect with each other. New segments can not make the width of B bigger than 1, thus the width of B is at most 1. From Lemma 4.2 and Lemma 4.3 we get rest of the stated thesis. □

Algorithm 6

```
Input:  $\sigma$, $\pi$ and $A$
Output: the number of solutions to $(\sigma, \pi)$ that respect $A$

begin
    for $i \in [m]$ do
        if there is no solution to $(\sigma[[l_i, r_i]], \pi[[l'_i, r'_i]])$ then
            return 0
        end
    end
    $B \leftarrow A$
    for $i \in [m]$ do
        if $r_i - l_i = r'_i - l'_i + 1$ then
            Compute constants $a$ and $b$ defined in Section 4.1 for $\sigma[[l_i, r_i]]$
              and $\pi[[l'_i, r'_i]]$
            Substitute $[l_i, r_i]$ with $[l_i, l_i + a - 1], [l_i + a, r_i - b], [r_i - b + 1, r_i]$
              in $B$
            Substitute $[l'_i, r'_i]$ with $[l'_i, l'_i + a - 1], [l'_i + a, r'_i - b - 1], [r'_i - b, r'_i]$
              in $B$
        end
    end
    for $i \in [k]$ do
        for $j$ in possible values of $\pi^{-1}[i]$ do
            $\mathbf{DP}_{i,j} \leftarrow \mathbf{Pref}_{i-1,j-1}$
        end
        Update $\mathbf{Pref}_i$ using $\mathbf{DP}_i$
    end
    return $\mathbf{Pref}_{k,n}$
end
```

Lemma 4.8. *Algorithm 6 computes the number of solutions to (σ, π) that respects A.*

Proof. Let f be one of the functions counted by Algorithm 6. We know that f fulfills:

1) all Y-axis constraints.

2) all X-axis constraints concerning two elements from two different segments $[l'_i, r'_i]$, since overlap of two segments is at most one element and we know that each element has to have different value, since Y-axis constraints are fulfilled.

3) all X-axis constraints in the i-th segment, where $r'_i = l'_i$, since there is exactly one element in this segment.

4) all X-axis constraints in the i-th segment, where $r_i - l_i = r'_i - l'_i$, since there is exactly one assignment from $[l'_i, r'_i]$ to $[l_i, r_i]$ that fulfills Y-axis constraints and there exists a solution to $(\sigma[[l_i, r_i]], \pi[[l'_i, r'_i]])$.

5) all X-axis constraints in the i-th segment, where $r_i - l_i = r'_i - l'_i + 1$. This follows from Lemma 4.4.

We obtained that each function f counted by Algorithm 6 fulfills all X-axis constraints, thus is a solution to (σ, π). Clearly, each solution to (σ, π) which respects A is counted by Algorithm 6, thus we obtain the stated thesis. $\square$

Lemma 4.9. *At most $2n$ entries in* $\mathbf{DP}$ *can take a nonzero value. Moreover, there are at most $2n$ entries in* $\mathbf{Pref}_{i,j}$*, such that* $\mathbf{Pref}_{i,j} \neq \mathbf{Pref}_{i,j-1}$.

Proof. For each $i \in [m]$ there are at most $2(r_i - l_i + 1)$ nonzero values in $\mathbf{DP}_{l'_i}, \ldots, \mathbf{DP}_{r'_i}$, since

case $r'_i = l'_i$**:** $\mathbf{DP}_{l'_i}$ can take at most $r_i - l_i + 1$ nonzero values,

case $r'_i - l'_i + 1 \geq r_i - l_i$**:** for $j \in [l'_i, r'_i]$ we know that $\mathbf{DP}_j$ can take at most two nonzero values.

As a result, the number of all nonzero values in $\mathbf{DP}$ is at most $\sum_{i=1}^{i=m} 2(r_i - l_i + 1) = 2n$. The second part of the lemma follows from the first part. $\square$

Lemma 4.10. *Algorithm 6 works in time $\mathcal{O}(n^2)$. Moreover, if $l'_i = r'_i$ for $i \in [m]$, then Algorithm 6 works in time $\mathcal{O}(n)$.*

Proof. If $(r_i - l_i + 1) - (r'_i - l'_i + 1) = 1$, then splitting the i-th segment of A in Algorithm 6 takes $\mathcal{O}((r_i - l_i + 1)^2)$ time, since for each position in segment $[l_i, r_i]$ we need to check if it induces a proper solution to Permutation Pattern Matching Problem. We conclude that splitting all segments takes in total $\mathcal{O}(\sum_{i\in[m]}(r_i - l_i + 1)^2) = \mathcal{O}(\sum_{i\in[m]}(r_i - l_i + 1) \cdot n) = \mathcal{O}(n^2)$ time.

From Lemma 4.9, its proof and the fact that Algorithm 6 computes values of $\mathbf{DP}$ and $\mathbf{Pref}$ in a monotonic way, we can compute value of $\mathbf{Pref}_{k,n}$ in $\mathcal{O}(n)$ time. $\square$

Lemma 4.11. *Algorithm 6 uses $\mathcal{O}(n)$ space.*

Proof. Space used by Algorithm 6 is proportional to the sum of the sizes of arrays $\mathbf{DP}$ and $\mathbf{Pref}$. From Lemma 4.9 and its proof we know that Algorithm 6 can store information about $\mathbf{DP}$ and $\mathbf{Pref}$ in $\mathcal{O}(n)$ space. This does not affect the time complexity, since Algorithm 6 computes the values of $\mathbf{DP}$ and $\mathbf{Pref}$ in a monotonic way. $\square$

4.3 Complement method

Let σ be a length-n permutation and π be a length-k permutation. Each solution f to (σ, π) can be interpreted as a set $A_f \subseteq [n]$ of size $n-k$, where $a \in A_f \Leftrightarrow a \notin f([k])$.

Let $m = \lfloor \frac{n-k}{2} \rfloor$. We call a set Z **sparse** if there is no integer i such that $i \in Z$ and $i+1 \in Z$. Let $\mathcal{C} = \{B \subseteq [2,n] : |B| = m \wedge B \text{ is } \textbf{sparse} \wedge (n \notin B \vee 2 \mid (n-k))\}$. For each $\{b_1, \dots, b_m\} = B \in \mathcal{C}$, we define S_B^n as a 1-proper segment decomposition of $[n]$ given by $l_i = b_{i-1} + 1$, $r_i = b_i - 1$ for $i \in [m]$, where $b_0 = 0$, additionally if $n \notin B$, then we define $l_{m+1} = b_m + 1$ and $r_{m+1} = n$. We denote length of S_B^n by d_B, which is equal to $m+1$ if $n \notin B$ and m otherwise.

Let $z = (n-k) \bmod 2$. For any $B \in \mathcal{C}$ we define a segment decomposition S_B^k of $[k]$ with length d_B as $l'_1 = 1$, $r'_i = l'_i + (r_i - l_i) - 1$ for $i \in [m]$ and $l'_{i+1} = r'_i + 1$ for $i \in [m-1]$, additionally if $d_B = m+1$, then we define $l'_{m+1} = r'_m + 1$ and $r'_{m+1} = l'_{m+1} + (r_{m+1} - l_{m+1}) - z$.

Lemma 4.12. *For any $B \in \mathcal{C}$, we know that S_B^k is a proper segment decomposition of $[k]$.*

Proof. We see that $r'_i + 1 = l'_{i+1}$ for $i \in [d_B - 1]$ and $l'_1 = 1$.

case $d_B = m+1$: $\sum_{i=1}^{i=m+1} r'_i - l'_i + 1 = (\sum_{i=1}^{i=m+1} r_i - l_i) + 1 - z = n - m - (m+1) + 1 - z = n - 2\lfloor \frac{n-k}{2} \rfloor - z = k$.

case $d_B = m$: $\sum_{i=1}^{i=m} r'_i - l'_i + 1 = \sum_{i=1}^{i=m} r_i - l_i = n - m - m = k$, since $d_B = m$ implies that $n \in B$, thus $2 \mid (n-k)$, so $2m = n-k$.

In both cases, we get that sum of lengths of all intervals in S_B^k is equal to k. □

For any $B \in \mathcal{C}$ we define S_B as a 1-proper segment decomposition of (σ, π) that consists of S_B^n and S_B^k.

Lemma 4.13. *For any $B \in \mathcal{C}$ the width of S_B is at most 1.*

Proof. $(r_i - l_i + 1) - (r'_i - l'_i + 1) = (r_i - l_i + 1) - ((l'_i + (r_i - l_i) - 1) - l'_i + 1) = 1$ for $i \in [m]$ and $(r_{m+1} - l_{m+1} + 1) - (r'_{m+1} - l'_{m+1} + 1) = (r_{m+1} - l_{m+1} + 1) - (l'_{m+1} + (r_{m+1} - l_{m+1}) - z - l'_{m+1} + 1) = z$. □

Lemma 4.14. *For each solution f to (σ, π) there exists exactly one $B \in \mathcal{C}$ such that f respects S_B.*

Proof. Let f be a solution to (σ, π), $\{b_1, \dots, b_m\} = B \in \mathcal{C}$ and $A_f = \{a_1, a_2, \dots, a_{n-k}\}$. We know that f respects S_B if and only if for each $i \in [0, m+z-1]$ there is exactly one value $c_i \in A_f \cap [b_i + 1, b_{i+1} - 1]$, where $b_0 = 0$ and if $z = 1$, then $b_{m+1} = n+1$. As a result f respects S_B if and only if $A_f = \{b_1, b_2, \dots, b_m\} \cup \{c_0, c_1, \dots, c_{m+z-1}\}$ and $b_i < c_i < b_{i+1}$ for $i \in [0, m+z-1]$, so f respects S_B if and only if $B = \{a_i : 2 \mid i\}$. See Figure 4.2 as an illustration. □

Theorem 4.15. *Counting Permutation Pattern Matching Problem can be solved in time $\mathcal{O}(n^2 \cdot \binom{\lceil \frac{n+k}{2} \rceil}{\lfloor \frac{n-k}{2} \rfloor})$ using $\mathcal{O}(n)$ space.*

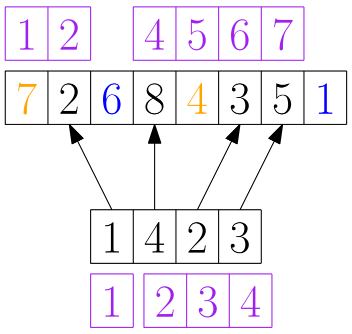


Figure 4.2: Situation from Lemma 4.14 for permutations $\sigma = (7,2,6,8,4,3,5,1)$, $\pi = (1,4,2,3)$, $c_0 = 1$, $c_1 = 5$, $B = \{3,8\}$ and a solution f to (σ,π), where $f(1) = 2$, $f(2) = 4$, $f(3) = 6$ and $f(4) = 7$. Black segments represent σ and π, arrows represent f. Orange values represents c_0 and c_1, blue values represents set B and purple segments represent S_B.

Algorithm 7

```
Input:  $\sigma, \pi$
Output: the number of solutions to $(\sigma, \pi)$

begin
    result ← 0
    foreach $B \in \mathcal{C}$ do
        result ← result + value returned by Algorithm 6 called with
          parameters $\sigma$, $\pi$, $S_B$
    end
    return result
end
```

Proof. From Lemma 4.14 we know that Algorithm 7 computes the number of solutions to (σ,π). $|\mathcal{C}| \le \binom{n-\lfloor\frac{n-k}{2}\rfloor}{\lfloor\frac{n-k}{2}\rfloor}$ and we can iterate over all sets in $\mathcal{C}$ in amortized time $\mathcal{O}(|\mathcal{C}|)$ using $\mathcal{O}(n)$ space, thus from Lemma 4.10 we know that Algorithm 7 works in time $\mathcal{O}(n^2 \cdot \binom{n-\lfloor\frac{n-k}{2}\rfloor}{\lfloor\frac{n-k}{2}\rfloor}) = \mathcal{O}(n^2 \cdot \binom{\lceil\frac{n+k}{2}\rceil}{\lfloor\frac{n-k}{2}\rfloor})$. From Lemma 4.11 we know that Algorithm 7 uses $\mathcal{O}(n)$ space. □

4.4 1-proper method

Let σ be a length-n permutation and π be a length-k permutation. Let $A = \{i : (2 \mid i) \wedge (1 \le i \le k)\}$, $B = \{i : (2 \mid i) \wedge (1 \le i \le n)\}$ and C be a set of all increasing functions $A \to B$.

For each $f \in C$ let S_f^n be a segment decomposition of $[n]$ with length k defined as $l_1 = 1$, $l_{2i} = f(2i)$, $r_{2i} = \min\{n, f(2i)+1\}$, $l_{2i+1} = r_{2i}$, $r_{2i+1} = l_{2i+2}$, additionally if $2 \nmid k$ then $r_k = n$. Let S_f^k be a proper segment decomposition of $[k]$ defined as

$l'_i = r'_i = i$ for $i \in [k]$. Let S_f be a segment decomposition of (σ, π) that consists of S_f^n and S_f^k. Let $\mathcal{D} = \{S_f : f \in C\}$.

Lemma 4.16. *For each $f \in C$ we know that S_f is a 1-proper segment decomposition of (σ, π) with width at most 1.*

Proof. The width of S_f is at most 1, since $l'_i = r'_i$. We see that $r_i \leq l_{i+1}$ for any $i \in [k-1]$, thus S_f^n is a 1-proper segment decomposition of $[n]$. □

Lemma 4.17. *For each solution h to (σ, π) there exists exactly one segment decomposition $D \in \mathcal{D}$ such that h respects D.*

Proof. For each solution h to (σ, π) we can construct f such that h respects S_f. We set $f(2i)$ to be a value, such that $h(2i)$ is one of the elements in the $2i$-th interval in S_f^n. There is a unique way to do it and h respects S_f. Our construction is the only one that is respected by h, thus we get the stated thesis. See Figure 4.3 as an illustration. □

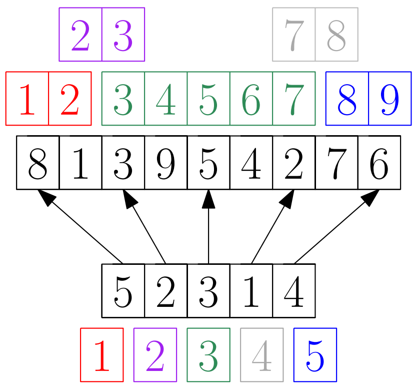


Figure 4.3: Situation from Lemma 4.17 for permutations $\sigma = (8, 1, 3, 9, 5, 4, 2, 7, 6)$, $\pi = (5, 2, 3, 1, 4)$ and a solution h to (σ, π), where $h(1) = 1$, $h(2) = 3$, $h(3) = 5$, $h(4) = 7$ and $h(5) = 9$. Arrows represent h, black segments represent σ and π. Red, purple, green, gray and blue segments represent S_f.

Theorem 4.18. *Counting Permutation Pattern Matching Problem can be solved in time $\mathcal{O}(n \cdot \binom{\lfloor \frac{n}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor})$ using $\mathcal{O}(n)$ space.*

Proof. From Lemma 4.17 we know that Algorithm 8 computes the number of solutions to (σ, π). Size of $\mathcal{D}$ is at most $\binom{\lfloor \frac{n}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}$. We can iterate over all elements in $\mathcal{D}$ in amortized time $\mathcal{O}(\binom{\lfloor \frac{n}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor})$ and $\mathcal{O}(n)$ space. For each $D \in \mathcal{D}$ Algorithm 8 runs Algorithm 6, so from Lemma 4.10 and Lemma 4.11 we get the stated thesis. □

Fact 4.19. $\binom{a}{b} \leq 2^a$ *for any* $0 \leq b \leq a$.

Theorem 4.20. *Counting Permutation Pattern Matching Problem can be solved in time $\mathcal{O}(n \cdot 2^{\lfloor \frac{n}{2} \rfloor})$ using $\mathcal{O}(n)$ space.*

Algorithm 8

```
Input:  $\sigma, \pi$
Output: the number of solutions to $(\sigma, \pi)$

begin
    result ← 0
    foreach $D \in \mathcal{D}$ do
        result ← result + value returned by Algorithm 6 called with
          parameters $\sigma, \pi, D$
    end
    return result
end
```

Proof. From Theorem 4.18 we know that there exists an algorithm that works in time $\mathcal{O}(n \cdot \binom{\lfloor \frac{n}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor})$ and uses $\mathcal{O}(n)$ space. From Fact 4.19 we know that $\mathcal{O}(n^2 \cdot \binom{\lfloor \frac{n}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}) = \mathcal{O}(n \cdot 2^{\lfloor \frac{n}{2} \rfloor})$, which concludes the proof. □

4.5 Lowerbound

For a fixed n and k let $\mathcal{D}_{n,k}$ be a set such that for each length-n permutation σ, length-k permutation π and a solution f to (σ, π) there exists exactly one $D \in \mathcal{D}_{n,k}$, such that D is a 1-proper segment decomposition of (σ, π) with width at most 1 and f respects D.

Lemma 4.21. *For any $D \in \mathcal{D}_{n,k}$ there exists a 1-proper segment decomposition A with width at most 1, such that for any length-n permutation σ, length-k permutation π and any solution f to (σ, π) we know that f respects D if and only if f respects A and segment decomposition of $[k]$ in A consists of segments with length one.*

Proof. For a pair $[l_i, r_i]$ and $[l'_i, r'_i]$, such that $r_i - l_i = r'_i - l'_i$ we substitute it with $r_i - l_i + 1$ pairs of segments $[j, j]$, $[l'_i - l_i + j, l'_i - l_i + j]$ for $j \in [l_i, r_i]$. For a pair $[l_i, r_i]$ and $[l'_i, r'_i]$, such that $r_i - l_i = r'_i - l'_i + 1$ we substitute it with $r_i - l_i + 1$ pairs of segments $[j, j+1]$, $[l'_i - l_i + j, l'_i - l_i + j]$ for $j \in [l_i, r_i - 1]$. We see that our construction fulfills all the requirements, thus it is the stated 1-proper segment decomposition. □

Corollary 4.22. *We can assume that $\mathcal{D}_{n,k}$ consists only of segment decompositions D, such that a segment decomposition of $[k]$ in D consists of segments with length one.*

Let $z = k \bmod 2$ and $m = 2 \cdot \lfloor \frac{n-z}{2} \rfloor$. Let C be a set of all increasing functions $f : [k-z] \to [m]$, such that $f(2i-1) = f(2i) - 1$ and $f(2i)$ is even for any $i \in [\frac{k-z}{2}]$. For each $f \in C$ we define a function $g_f : [k] \to [n]$ by $g_f(i) = f(i)$ for $i \in [k-z]$ and if $z = 1$, then $g_f(k) = n$. Let B be a set of all increasing functions $[\frac{k-z}{2}] \to [\frac{m}{2}]$.

Observation 4.23. $\lfloor \frac{k}{2} \rfloor = \frac{k-z}{2}$, $\frac{m}{2} \geq \lfloor \frac{n-1}{2} \rfloor$

Lemma 4.24. $|C| \geq \binom{\lfloor \frac{n-1}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}$

Proof. We know that $k-z$ and m are even numbers, thus for each function $h \in B$ exists a function $f_h \in C$, such that $f_h(2i-1) = 2h(i)-1$ and $f_h(2i) = 2h(i)$. Additionally, for any $h, h' \in B$ if $f_h = f_{h'}$, then $h = h'$, thus $|C| \geq |B| = \binom{\frac{m}{2}}{\frac{k-z}{2}} = \binom{\frac{m}{2}}{\lfloor \frac{k}{2} \rfloor} \geq \binom{\lfloor \frac{n-1}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}$, where last two transitions follow from Observation 4.23. $\square$

Fact 4.25. *For any function $f : [k] \to [n]$, there exists length-n permutation σ and length-k permutation π, such that f is a solution to (σ, π).*

Lemma 4.26. $|\mathcal{D}_{n,k}| \geq \binom{\lfloor \frac{n-1}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}$

Proof. From Fact 4.25 and property of $\mathcal{D}_{n,k}$ for any $f \in C$ exists exactly one $D_f \in \mathcal{D}_{n,k}$, such that g_f respects D_f. Let $[l_1, r_1], \ldots, [l_k, r_k]$ be a segment decomposition of $[n]$ that D_f consists of. Since $g(2i-1) = f(2i-1) = f(2i)-1 = g(2i)-1$ we know that $f(2i-1) \leq p_{2i-1} \leq l_{2i} \leq f(2i) = f(2i-1)+1$, thus $p_{2i-1} \in \{f(2i-1), f(2i)\}$. As a result, for any $f, f' \in C$ if $D_f = D_{f'}$, then $f = f'$, thus $|\mathcal{D}_{n,k}| \geq |C| \geq \binom{\lfloor \frac{n-1}{2} \rfloor}{\lfloor \frac{k}{2} \rfloor}$, where the last inequality follows from Lemma 4.24. $\square$

Corollary 4.27. *Family of 1-proper segment decompositions with width at most* 1 *presented in Section 4.4 has a size that is very close to the optimal.*

Bibliography

[1] Shlomo Ahal and Yuri Rabinovich. On complexity of the subpattern problem. *SIAM J. Discret. Math.*, 22(2):629–649, 2008.

[2] Michael H. Albert, Robert E. L. Aldred, Mike D. Atkinson, and Derek A. Holton. Algorithms for pattern involvement in permutations. In Peter Eades and Tadao Takaoka, editors, *Algorithms and Computation*, pages 355–367, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[3] Benjamin Aram Berendsohn, László Kozma, and Dániel Marx. Finding and Counting Permutations via CSPs. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[4] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern matching for permutations. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, Nicola Santoro, and Sue Whitesides, editors, *Algorithms and Data Structures, Third Workshop, WADS '93, Montréal, Canada, August 11-13, 1993, Proceedings*, volume 709 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 1993.

[5] Marie-Louise Bruner and Martin Lackner. A fast algorithm for permutation pattern matching based on alternating runs. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory – SWAT 2012*, pages 261–270, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[6] Bartłomiej Dudek and Paweł Gawrychowski. Counting 4-Patterns in Permutations Is Equivalent to Counting 4-Cycles in Graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[7] Fedor V. Fomin, Serge Gaspers, Saket Saurabh, and Alexey A. Stepanov. On two techniques of combining branching and treewidth. *Algorithmica*, 54(2):181–207, 2009.

[8] Jacob Fox. Stanley-wilf limits are typically exponential. *CoRR*, abs/1310.8378, 2013.

[9] Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 82–101. SIAM, 2014.

[10] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[11] Adam Marcus and Gábor Tardos. Excluded permutation matrices and the stanley-wilf conjecture. *J. Comb. Theory, Ser. A*, 107(1):153–160, 2004.