Order-preserving pattern matching with k mismatches

Paweł Gawrychowski[1] and Przemysław Uznański[2]

Max-Planck-Institut für Informatik, Saarbrücken, Germany

LIF, CNRS and Aix-Marseille Université, Marseille, France

The starting point

Order-preserving pattern matching

Given a text, find its fragment, which is order-isomorphic to the pattern.

Two sequences of numbers are order-isomorphic if their sorting permutations are the same:

$$(4, 1, 2, 3) \sim (63, 12, 23, 42).$$

Order-preserving pattern matching

Input: text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.
Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \sim (p_1, p_2, \ldots, p_m)$?

A closely related (but different) question is that of locating a permutation pattern, which is a **subsequence** of the text with the same sorting permutation as the one of the pattern.

The starting point

Order-preserving pattern matching

Given a text, find its fragment, which is order-isomorphic to the pattern.

Two sequences of numbers are order-isomorphic if their sorting permutations are the same:

$$(4, 1, 2, 3) \sim (63, 12, 23, 42).$$

Order-preserving pattern matching

Input: text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.
Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \sim (p_1, p_2, \ldots, p_m)$?

A closely related (but different) question is that of locating a permutation pattern, which is a **subsequence** of the text with the same sorting permutation as the one of the pattern.

The starting point

Order-preserving pattern matching

Given a text, find its fragment, which is order-isomorphic to the pattern.

Two sequences of numbers are order-isomorphic if their sorting permutations are the same:

$$(4, 1, 2, 3) \sim (63, 12, 23, 42).$$

Order-preserving pattern matching

Input: text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.
Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \sim (p_1, p_2, \ldots, p_m)$?

A closely related (but different) question is that of locating a permutation pattern, which is a **subsequence** of the text with the same sorting permutation as the one of the pattern.

The starting point

Order-preserving pattern matching

Given a text, find its fragment, which is order-isomorphic to the pattern.

Two sequences of numbers are order-isomorphic if their sorting permutations are the same:

$$(4, 1, 2, 3) \sim (63, 12, 23, 42).$$

Order-preserving pattern matching

Input: text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.
Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \sim (p_1, p_2, \ldots, p_m)$?

A closely related (but different) question is that of locating a permutation pattern, which is a **subsequence** of the text with the same sorting permutation as the one of the pattern.

Previous results

Quite a few papers last/this year. The final message?

Order-preserving pattern matching can be solved in time $\mathcal{O}(n + \mathrm{sort}(m))$.

So what should be the next step?

Previous results

Quite a few papers last/this year. The final message?

Order-preserving pattern matching can be solved in time $\mathcal{O}(n + \mathrm{sort}(m))$.

So what should be the next step?

Previous results

Quite a few papers last/this year. The final message?

Order-preserving pattern matching can be solved in time $\mathcal{O}(n + \mathrm{sort}(m))$.

So what should be the next step?

The next step

What about an approximate variant? In other words, what would it mean to find a fragment of the text which is **almost** order-isomorphic to the pattern?

For the usual pattern matching, there are two natural definitions of what almost could mean:

1. pattern matching with k mismatches,
2. pattern matching with k errors.

The next step

What about an approximate variant? In other words, what would it mean to find a fragment of the text which is **almost** order-isomorphic to the pattern?

For the usual pattern matching, there are two natural definitions of what almost could mean:

1. pattern matching with k mismatches,
2. pattern matching with k errors.

The next step

What about an approximate variant? In other words, what would it mean to find a fragment of the text which is **almost** order-isomorphic to the pattern?

For the usual pattern matching, there are two natural definitions of what almost could mean:

1. pattern matching with k mismatches,
2. pattern matching with k errors.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, remove the numbers at the corresponding positions from both sequences, and get order-isomorphic sequences.

$$(1, 4, 2, 5, 11) \overset{1}{\sim} (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, remove the numbers at the corresponding positions from both sequences, and get order-isomorphic sequences.

$$(1, 4, 2, 5, 11) \overset{1}{\sim} (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, remove the numbers at the corresponding positions from both sequences, and get order-isomorphic sequences.

$$(1, 4, 2, 5, 11) \overset{1}{\sim} (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, remove the numbers at the corresponding positions from both sequences, and get order-isomorphic sequences.

$$(1, 4, 2, 11) \sim (4, 8, 5, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, modify the numbers at the corresponding positions in the first sequence, and get order-isomorphic sequences.

$$(1, 4, 2, 5, 11) \overset{1}{\sim} (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, modify the numbers at the corresponding positions in the first sequence, and get order-isomorphic sequences.

$$(1, 4, 2, 5, 11) \overset{1}{\sim} (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, modify the numbers at the corresponding positions in the first sequence, and get order-isomorphic sequences.

$$(1, 4, 2, 3, 11) \sim (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, modify the numbers at the corresponding positions in the first sequence, and get order-isomorphic sequences.

$$(1, 4, 2, 3, 11) \sim (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, modify the numbers at the corresponding positions in the first sequence, and get order-isomorphic sequences.

$$(1, 4, 2, 3, 11) \sim (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Order-preserving pattern matching with k mismatches

$(a_1, a_2, \ldots, a_m) \overset{k}{\sim} (b_1, b_2, \ldots, b_m)$ if we can choose up to k indices $i_1 < i_2 < \ldots < i_k$, modify the numbers at the corresponding positions in the first sequence, and get order-isomorphic sequences.

$$(1, 4, 2, 3, 11) \sim (4, 8, 5, 7, 9)$$

Order-preserving pattern matching with k mismatches

Input: number k, text $(t_1, t_2, \ldots, t_n)$ and pattern $(p_1, p_2, \ldots, p_m)$.

Output: is there i such that $(t_i, t_{i+1}, \ldots, t_{i+m-1}) \overset{k}{\sim} (p_1, p_2, \ldots, p_m)$?

$(1, 4, 2, 5, 11)$ occurs in $(1, 10, 6, 4, 8, 5, 7, 9, 3)$ with 1 mismatch.

Simplifying assumption

The numbers (both in the text and the pattern) don't repeat.

How to check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$?

A very simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ iff there exist $i_1, i_2, \ldots, i_{m-k}$ such that $a_{i_1} < a_{i_2} < \ldots < a_{i_{m-k}}$ and $b_{i_1} < b_{i_2} < \ldots < b_{i_{m-k}}$.

Simplifying assumption

The numbers (both in the text and the pattern) don't repeat.

How to check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$?

A very simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ iff there exist $i_1, i_2, \ldots, i_{m-k}$ such that $a_{i_1} < a_{i_2} < \ldots < a_{i_{m-k}}$ and $b_{i_1} < b_{i_2} < \ldots < b_{i_{m-k}}$.

Simplifying assumption

The numbers (both in the text and the pattern) don't repeat.

How to check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$?

A very simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ iff there exist $i_1, i_2, \ldots, i_{m-k}$ such that $a_{i_1} < a_{i_2} < \ldots < a_{i_{m-k}}$ and $b_{i_1} < b_{i_2} < \ldots < b_{i_{m-k}}$.

Using the simple lemma, we can check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$.

$$(42, 54, 23, 9, 25, 15, 21, 10, 51, 63) \overset{3}{\sim} (20, 23, 10, 4, 16, 8, 14, 1, 40, 46)$$

1. Rearrange the indices in both sequences so that $(b_1, \ldots, b_m)$ is increasing.
2. Then check if $(a_1, \ldots, a_m)$ contains an increasing subsequence of length $m - k$.

A simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ can be verified in $\mathcal{O}(m \log \log m)$ time.

Using the simple lemma, we can check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$.

$$(42, 54, 23, 9, 25, 15, 21, 10, 51, 63) \overset{3}{\sim} (20, 23, 10, 4, 16, 8, 14, 1, 40, 46)$$

1. Rearrange the indices in both sequences so that $(b_1, \ldots, b_m)$ is increasing.
2. Then check if $(a_1, \ldots, a_m)$ contains an increasing subsequence of length $m - k$.

A simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ can be verified in $O(m \log \log m)$ time.

Using the simple lemma, we can check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$.

$$(42, 54, 23, 9, 25, 15, 21, 10, 51, 63) \overset{3}{\sim} (20, 23, 10, 4, 16, 8, 14, 1, 40, 46)$$

1. Rearrange the indices in both sequences so that $(b_1, \ldots, b_m)$ is increasing.
2. Then check if $(a_1, \ldots, a_m)$ contains an increasing subsequence of length $m - k$.

A simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ can be verified in $\mathcal{O}(m \log \log m)$ time.

Using the simple lemma, we can check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$.

$$(10, 9, 15, 23, 21, 25, 42, 54, 51, 63) \overset{3}{\sim} (1, 4, 8, 10, 14, 16, 20, 23, 40, 46)$$

1. Rearrange the indices in both sequences so that $(b_1, \ldots, b_m)$ is increasing.
2. Then check if $(a_1, \ldots, a_m)$ contains an increasing subsequence of length $m - k$.

A simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ can be verified in $\mathcal{O}(m \log \log m)$ time.

Using the simple lemma, we can check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$.

$$(10, 9, 15, 23, 21, 25, 42, 54, 51, 63) \overset{3}{\sim} (1, 4, 8, 10, 14, 16, 20, 23, 40, 46)$$

1. Rearrange the indices in both sequences so that $(b_1, \ldots, b_m)$ is increasing.
2. Then check if $(a_1, \ldots, a_m)$ contains an increasing subsequence of length $m - k$.

A simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ can be verified in $\mathcal{O}(m \log \log m)$ time.

Using the simple lemma, we can check if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$.

$$(10, 9, 15, 23, 21, 25, 42, 54, 51, 63) \overset{3}{\sim} (1, 4, 8, 10, 14, 16, 20, 23, 40, 46)$$

1. Rearrange the indices in both sequences so that $(b_1, \ldots, b_m)$ is increasing.
2. Then check if $(a_1, \ldots, a_m)$ contains an increasing subsequence of length $m - k$.

A simple lemma

$(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ can be verified in $\mathcal{O}(m \log \log m)$ time.

By iterating over all possible starting positions, we get a simple $\mathcal{O}(nm \log \log m)$ time solution.

But this is a boring answer. The usual assumption is that k is small, and the goal is to achieve $\mathcal{O}(nf(k))$ time complexity, where $f(k)$ is some (hopefully slowly growing) function of k.

By iterating over all possible starting positions, we get a simple $\mathcal{O}(nm \log \log m)$ time solution.

But this is a boring answer. The usual assumption is that k is small, and the goal is to achieve $\mathcal{O}(nf(k))$ time complexity, where $f(k)$ is some (hopefully slowly growing) function of k.

Plan of the attack

We move a window of length m over the text. For every possible alignment, we either:

1. quickly detect that the number of mismatches must necessarily exceed k, or
2. figure out that the current window is quite similar to the pattern, and use the similarity to speed up checking if the number of mismatches is indeed at most k.

Such high-level idea has been used in both the “usual” approximate pattern matching, and the so-called parametrised approximate pattern matching.

Plan of the attack

We move a window of length m over the text. For every possible alignment, we either:

1. quickly detect that the number of mismatches must necessarily exceed k, or
2. figure out that the current window is quite similar to the pattern, and use the similarity to speed up checking if the number of mismatches is indeed at most k.

Such high-level idea has been used in both the “usual” approximate pattern matching, and the so-called parametrised approximate pattern matching.

Plan of the attack

We move a window of length m over the text. For every possible alignment, we either:

1. quickly detect that the number of mismatches must necessarily exceed k, or
2. figure out that the current window is quite similar to the pattern, and use the similarity to speed up checking if the number of mismatches is indeed at most k.

Such high-level idea has been used in both the “usual” approximate pattern matching, and the so-called parametrised approximate pattern matching.

Plan of the attack

We move a window of length m over the text. For every possible alignment, we either:

1. quickly detect that the number of mismatches must necessarily exceed k, or
2. figure out that the current window is quite similar to the pattern, and use the similarity to speed up checking if the number of mismatches is indeed at most k.

Such high-level idea has been used in both the “usual” approximate pattern matching, and the so-called parametrised approximate pattern matching.

Signatures

To quickly eliminate some starting positions, we use the notion of a signature. The intuition is that our signature captures **some** of the order-structure, but doesn't change much when we move the window.

$S(a_1, \ldots, a_m)$

For every i we find the predecessor of a_i in the whole $\{a_1, a_2, \ldots, a_m\}$ and denote by $\mathrm{pred}(i)$ the place where this predecessor occurs in the sequence. (If there is no predecessor, $\mathrm{pred}(i) = 0$.)

$$S(a_1, \ldots, a_m) = (1 - \mathrm{pred}(1), \ldots, m - \mathrm{pred}(m))$$

$$\begin{aligned} S(11,\ 4,\ 12,\ 1,\ 9,\ 3,\ 10,\ 7,\ 2,\ 5,\ 13,\ 0,\ 6,\ 8) = \\ (6,\ 4,\ -2,\ 8,\ 9,\ 3,\ -2,\ 5,\ -5,\ -8,\ -8,\ 0,\ -3,\ -6) \end{aligned}$$

Signatures

To quickly eliminate some starting positions, we use the notion of a signature. The intuition is that our signature captures **some** of the order-structure, but doesn't change much when we move the window.

$S(a_1, \ldots, a_m)$

For every i we find the predecessor of a_i in the whole $\{a_1, a_2, \ldots, a_m\}$ and denote by $\mathrm{pred}(i)$ the place where this predecessor occurs in the sequence. (If there is no predecessor, $\mathrm{pred}(i) = 0$.)

$$S(a_1, \ldots, a_m) = (1 - \mathrm{pred}(1), \ldots, m - \mathrm{pred}(m))$$

$$S(11,\ 4,\ 12,\ 1,\ 9,\ 3,\ 10,\ 7,\ 2,\ 5,\ 13,\ 0,\ 6,\ 8) =$$
$$(6,\ 4,\ -2,\ 8,\ 9,\ 3,\ -2,\ 5,\ -5,\ -8,\ -8,\ 0,\ -3,\ -6)$$

Signatures

To quickly eliminate some starting positions, we use the notion of a signature. The intuition is that our signature captures **some** of the order-structure, but doesn't change much when we move the window.

$S(a_1, \ldots, a_m)$

For every i we find the predecessor of a_i in the whole $\{a_1, a_2, \ldots, a_m\}$ and denote by $\mathrm{pred}(i)$ the place where this predecessor occurs in the sequence. (If there is no predecessor, $\mathrm{pred}(i) = 0$.)

$$S(a_1, \ldots, a_m) = (1 - \mathrm{pred}(1), \ldots, m - \mathrm{pred}(m))$$

$$\begin{aligned} S(&11, 4, 12, 1, 9, 3, 10, 7, 2, 5, 13, 0, 6, 8) = \\ &(6, 4, -2, 8, 9, 3, -2, 5, -5, -8, -8, 0, -3, -6) \end{aligned}$$

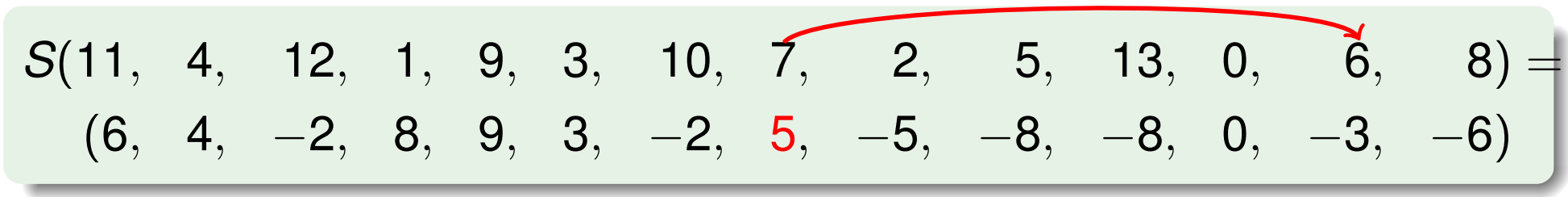
Signatures

To quickly eliminate some starting positions, we use the notion of a signature. The intuition is that our signature captures **some** of the order-structure, but doesn't change much when we move the window.

$S(a_1, \ldots, a_m)$

For every i we find the predecessor of a_i in the whole $\{a_1, a_2, \ldots, a_m\}$ and denote by $\mathrm{pred}(i)$ the place where this predecessor occurs in the sequence. (If there is no predecessor, $\mathrm{pred}(i) = 0$.)

$$S(a_1, \ldots, a_m) = (1 - \mathrm{pred}(1), \ldots, m - \mathrm{pred}(m))$$



First crucial property of signatures

Lemma

If $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ then the Hamming distance between $S(a_1, \ldots, a_m)$ and $S(b_1, \ldots, b_m)$ is at most $3k$.

Proof: induction on k.

$$
\begin{aligned}
S(11, 4, 12, 1, 9, 3, 10, 7, 2, 5, 13, 0, 6, 8) &= (6, \ 4, -2, \ 8, 9, 3, -2, 5, -5, -8, -8, 0, -3, -6) \\
S(10, 1, 11, 2, 9, 4, 12, 7, 3, 5, 13, 0, 6, 8) &= (4, 10, -2, -2, 9, 3, -4, 5, -5, -4, -4, 0, -3, -6).
\end{aligned}
$$

The sequences are 2-isomorphic and the Hamming distance is 6.

First crucial property of signatures

Lemma

If $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ then the Hamming distance between $S(a_1, \ldots, a_m)$ and $S(b_1, \ldots, b_m)$ is at most $3k$.

Proof: induction on k.

$$S(11, 4, 12, 1, 9, 3, 10, 7, 2, 5, 13, 0, 6, 8) = (6, 4, -2, 8, 9, 3, -2, 5, -5, -8, -8, 0, -3, -6)$$
$$S(10, 1, 11, 2, 9, 4, 12, 7, 3, 5, 13, 0, 6, 8) = (4, 10, -2, -2, 9, 3, -4, 5, -5, -4, -4, 0, -3, -6).$$

The sequences are 2-isomorphic and the Hamming distance is 6.

First crucial property of signatures

Lemma

If $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ then the Hamming distance between $S(a_1, \ldots, a_m)$ and $S(b_1, \ldots, b_m)$ is at most $3k$.

Proof: induction on k.

$$
\begin{aligned}
S(11, 4, 12, 1, 9, 3, 10, 7, 2, 5, 13, 0, 6, 8) &= (6, \ 4, -2, \ 8, 9, 3, -2, 5, -5, -8, -8, 0, -3, -6) \\
S(10, 1, 11, 2, 9, 4, 12, 7, 3, 5, 13, 0, 6, 8) &= (4, 10, -2, -2, 9, 3, -4, 5, -5, -4, -4, 0, -3, -6).
\end{aligned}
$$

The sequences are 2-isomorphic and the Hamming distance is 6.

Second crucial property of signatures

Recall that we move a window of length m over the text. It turns out that the corresponding signature $S(t_i, \ldots, t_{i+m-1})$ doesn't change very much.

Updating the signature

If $S(t_i, \ldots, t_{i+m-1}) = (s_1, \ldots, s_m)$, then to create $S(t_{i+1}, \ldots, t_{i+m})$ we only need to:

1. remove s_1 from the beginning,
2. add a new s_{m+1} in the end,
3. replace at most two characters of the resulting string.

Second crucial property of signatures

Recall that we move a window of length m over the text. It turns out that the corresponding signature $S(t_i, \ldots, t_{i+m-1})$ doesn't change very much.

Updating the signature

If $S(t_i, \ldots, t_{i+m-1}) = (s_1, \ldots, s_m)$, then to create $S(t_{i+1}, \ldots, t_{i+m})$ we only need to:

1. remove s_1 from the beginning,
2. add a new s_{m+1} in the end,
3. replace at most two characters of the resulting string.

The first part of the plan

We maintain the signature as we move the window over the text. At every position we want to check if the Hamming distance between the current signature and $S(p_1, \ldots, p_m)$ is at most $3k$. If not, there are more than k mismatches!

But how to implement the check efficiently?

Use a few standard tools (suffix array, lcp queries, ...)

The first part of the plan

We maintain the signature as we move the window over the text. At every position we want to check if the Hamming distance between the current signature and $S(p_1, \ldots, p_m)$ is at most $3k$. If not, there are more than k mismatches!

But how to implement the check efficiently?

Use a few standard tools (suffix array, lcp queries, ...)

The first part of the plan

We maintain the signature as we move the window over the text. At every position we want to check if the Hamming distance between the current signature and $S(p_1, \ldots, p_m)$ is at most $3k$. If not, there are more than k mismatches!

But how to implement the check efficiently?

Use a few standard tools (suffix array, lcp queries, ...)

The second part of the plan

Now we know that the Hamming distance between $S(p_1, \ldots, p_m)$ and $S(t_i, \ldots, t_{i+m-1})$ is $\leq k$. How to check if $(p_1, \ldots, p_m) \overset{k}{\sim} (t_i, \ldots, t_{i+m-1})$ in time depending on k instead of m?

We will reduce it to the following problem.

Heaviest increasing subsequence

Input: $(a_1, a_2, \ldots, a_m)$ and weight w_i of every a_i.
Output: increasing subsequence with the largest total weight.

Very similar to the longest increasing subsequence. Can be solved in the same complexity.

The second part of the plan

Now we know that the Hamming distance between $S(p_1, \ldots, p_m)$ and $S(t_i, \ldots, t_{i+m-1})$ is $\leq k$. How to check if $(p_1, \ldots, p_m) \overset{k}{\sim} (t_i, \ldots, t_{i+m-1})$ in time depending on k instead of m?
We will reduce it to the following problem.

Heaviest increasing subsequence

Input: $(a_1, a_2, \ldots, a_m)$ and weight w_i of every a_i.
Output: increasing subsequence with the largest total weight.

Very similar to the longest increasing subsequence. Can be solved in the same complexity.

The second part of the plan

Now we know that the Hamming distance between $S(p_1, \ldots, p_m)$ and $S(t_i, \ldots, t_{i+m-1})$ is $\leq k$. How to check if $(p_1, \ldots, p_m) \overset{k}{\sim} (t_i, \ldots, t_{i+m-1})$ in time depending on k instead of m?
We will reduce it to the following problem.

Heaviest increasing subsequence

Input: $(a_1, a_2, \ldots, a_m)$ and weight w_i of every a_i.
Output: increasing subsequence with the largest total weight.

Very similar to the longest increasing subsequence. Can be solved in the same complexity.

Final piece of the puzzle

If the signatures of two sequences agree on most positions, then checking if they are order-isomorphic with k mismatches is easy.

Lemma

Given ℓ positions where $S(a_1, \ldots, a_m)$ and $S(b_1, \ldots, b_m)$ differ, we can reduce in $\mathcal{O}(\ell \log\log \ell)$ time checking if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ to computing the heaviest increasing subsequence on at most $\ell + 1$ elements.

...assuming random access to $(a_1, \ldots, a_m)$, the sorting permutation π_b of $(b_1, \ldots, b_m)$ and the rank of every b_i in $\{b_1, \ldots, b_m\}$.

Final piece of the puzzle

If the signatures of two sequences agree on most positions, then checking if they are order-isomorphic with k mismatches is easy.

Lemma

Given ℓ positions where $S(a_1, \ldots, a_m)$ and $S(b_1, \ldots, b_m)$ differ, we can reduce in $\mathcal{O}(\ell \log\log \ell)$ time checking if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ to computing the heaviest increasing subsequence on at most $\ell + 1$ elements.

...assuming random access to $(a_1, \ldots, a_m)$, the sorting permutation π_b of $(b_1, \ldots, b_m)$ and the rank of every b_i in $\{b_1, \ldots, b_m\}$.

Final piece of the puzzle

If the signatures of two sequences agree on most positions, then checking if they are order-isomorphic with k mismatches is easy.

Lemma

Given ℓ positions where $S(a_1, \ldots, a_m)$ and $S(b_1, \ldots, b_m)$ differ, we can reduce in $\mathcal{O}(\ell \log\log \ell)$ time checking if $(a_1, \ldots, a_m) \overset{k}{\sim} (b_1, \ldots, b_m)$ to computing the heaviest increasing subsequence on at most $\ell + 1$ elements.

...assuming random access to $(a_1, \ldots, a_m)$, the sorting permutation π_b of $(b_1, \ldots, b_m)$ and the rank of every b_i in $\{b_1, \ldots, b_m\}$.

What?!

Recall that what we really want is to find $i_1, i_2, \ldots, i_{m-k}$ such that $a_{i_1} < \ldots < a_{i_{m-k}}$ and $b_{i_1} < \ldots < b_{i_{m-k}}$.

If there is no mismatch between the i-th characters of both signatures, then the predecessors of a_i and b_i in their respective sequences are at the same position j. Then either we should take both i and j in our solution, or neither of them.

So the only decisions we need to make concern positions i such that the signatures differ there, and there are just ℓ such positions.

What?!

Recall that what we really want is to find $i_1, i_2, \ldots, i_{m-k}$ such that $a_{i_1} < \ldots < a_{i_{m-k}}$ and $b_{i_1} < \ldots < b_{i_{m-k}}$.

If there is no mismatch between the i-th characters of both signatures, then the predecessors of a_i and b_i in their respective sequences are at the same position j. Then either we should take both i and j in our solution, or neither of them.

So the only decisions we need to make concern positions i such that the signatures differ there, and there are just ℓ such positions.

What?!

Recall that what we really want is to find $i_1, i_2, \ldots, i_{m-k}$ such that $a_{i_1} < \ldots < a_{i_{m-k}}$ and $b_{i_1} < \ldots < b_{i_{m-k}}$.

If there is no mismatch between the i-th characters of both signatures, then the predecessors of a_i and b_i in their respective sequences are at the same position j. Then either we should take both i and j in our solution, or neither of them.

So the only decisions we need to make concern positions i such that the signatures differ there, and there are just ℓ such positions.

By combining all these ingredients, we process every possible starting position in $\mathcal{O}(\log\log n + k\log\log k)$ time.

Final result

Order-preserving pattern matching with k mismatches can be solved in $\mathcal{O}(n(\log\log m + k\log\log k))$ time.

By combining all these ingredients, we process every possible starting position in $\mathcal{O}(\log\log n + k\log\log k)$ time.

Final result

Order-preserving pattern matching with k mismatches can be solved in $\mathcal{O}(n(\log\log m + k\log\log k))$ time.

An open problem

A natural question is whether we can solve order-preserving pattern matching with k errors efficiently.

Order-isomorphism with k errors

Two sequences $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ are order-isomorphic with k errors if we can remove up to k elements from each of them, not necessarily at the same positions, and get two order-isomorphic sequences.

Can you construct an $\mathcal{O}(nf(k))$ time algorithm, where $f(k)$ is **any** function of k?

An open problem

A natural question is whether we can solve order-preserving pattern matching with k errors efficiently.

Order-isomorphism with k errors

Two sequences $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ are order-isomorphic with k errors if we can remove up to k elements from each of them, not necessarily at the same positions, and get two order-isomorphic sequences.

Can you construct an $\mathcal{O}(nf(k))$ time algorithm, where $f(k)$ is **any** function of k?

An open problem

A natural question is whether we can solve order-preserving pattern matching with k errors efficiently.

Order-isomorphism with k errors

Two sequences $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ are order-isomorphic with k errors if we can remove up to k elements from each of them, not necessarily at the same positions, and get two order-isomorphic sequences.

Can you construct an $\mathcal{O}(nf(k))$ time algorithm, where $f(k)$ is **any** function of k?

Another open problem

Can you decrease the complexity to, say, $\tilde{\mathcal{O}}(n\sqrt{k})$ with more combinatorial insight?