Structural analysis of approximate pattern matching and algorithmic applications

Panagiotis Charalampopoulos

Birkbeck, University of London, UK

Summer School, École normale supérieure

25 June 2023

Roadmap

Roadmap

Periodicity

Roadmap

Periodicity

Approximate pattern matching under Hamming and edit distance

- Structural results: either the pattern is almost periodic or it only has a few approximate occurrences.
- A fast algorithm that relies on primitive (PILLAR) operations.

Roadmap

Periodicity

Approximate pattern matching under Hamming and edit distance

- Structural results: either the pattern is almost periodic or it only has a few approximate occurrences.
- A fast algorithm that relies on primitive (PILLAR) operations.

Implementations of PILLAR operations in different settings: the standard setting and the compressed setting.

Roadmap

Periodicity

Approximate pattern matching under Hamming and edit distance

- Structural results: either the pattern is almost periodic or it only has a few approximate occurrences.
- A fast algorithm that relies on primitive (PILLAR) operations.

Implementations of PILLAR operations in different settings: the standard setting and the compressed setting.

Further improvements to the obtained algorithm for approximate pattern matching under edit distance.

Periodicity

Periodicity

Periodicity is one of the most elegant notions in algorithms and combinatorics on strings.

Periodicity

Periodicity is one of the most elegant notions in algorithms and combinatorics on strings.

An integer $p > 0$ is a period of a string S if $S[i] = S[i+p]$ for all $i = 1, \ldots, |S| - p$.

Periodicity

Periodicity is one of the most elegant notions in algorithms and combinatorics on strings.

An integer $p > 0$ is a period of a string S if $S[i] = S[i + p]$ for all $i = 1, \ldots, |S| - p$.

The smallest period of string S is the period of S, and is denoted by $\text{per}(S)$.

Periodicity

Periodicity is one of the most elegant notions in algorithms and combinatorics on strings.

An integer $p > 0$ is a period of a string S if $S[i] = S[i+p]$ for all $i = 1, \ldots, |S| - p$.

The smallest period of string S is the period of S, and is denoted by $\mathrm{per}(S)$.

For example, the period of $S = a\,b\,c\,a\,b\,c\,a\,b\,c\,a\,b$ is 3.

Periodicity

Periodicity is one of the most elegant notions in algorithms and combinatorics on strings.

An integer $p > 0$ is a period of a string S if $S[i] = S[i+p]$ for all $i = 1, \ldots, |S| - p$.

The smallest period of string S is the period of S, and is denoted by $\mathrm{per}(S)$.

For example, the period of $S = a\,b\,c\,a\,b\,c\,a\,b\,c\,a\,b$ is 3.

6 and 9 are also periods of S.

Periodicity

Periodicity is one of the most elegant notions in algorithms and combinatorics on strings.

An integer $p > 0$ is a period of a string S if $S[i] = S[i+p]$ for all $i = 1, \ldots, |S| - p$.

The smallest period of string S is the period of S, and is denoted by $\mathrm{per}(S)$.

For example, the period of $S = a\,b\,c\,a\,b\,c\,a\,b\,c\,a\,b$ is 3.

6 and 9 are also periods of S.

We say that a string S is periodic if $\mathrm{per}(S) \leq |S|/2$.

The periodicity lemma (weak version)

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf '65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf '65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

Proof. We show that $q - p$ is a period of S.

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf ’65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

Proof. We show that $q - p$ is a period of S.

Let $j = i + q - p$.

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf '65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

Proof. We show that $q - p$ is a period of S.

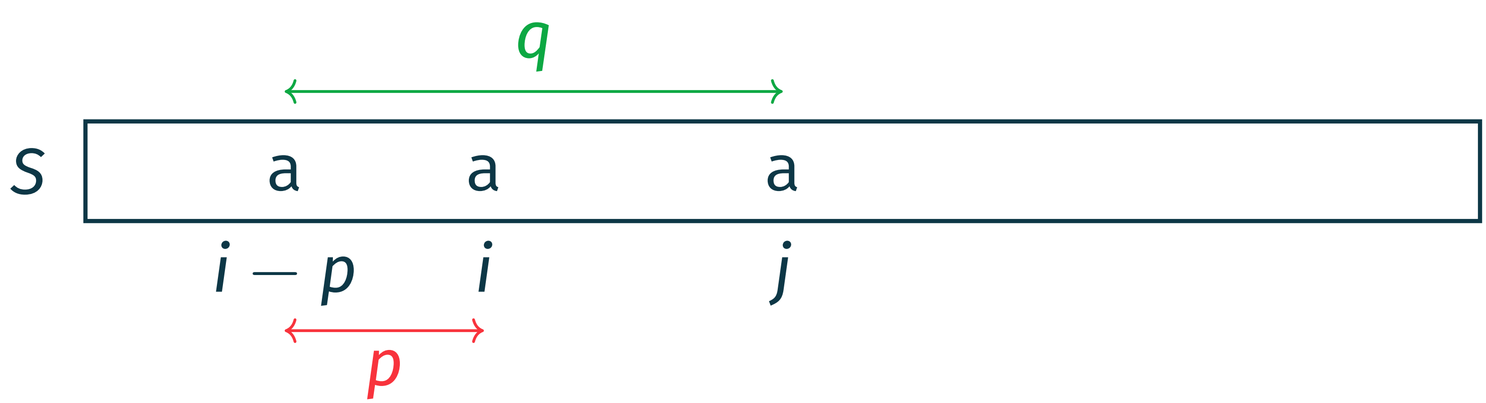
Let $j = i + q - p$.

$p + q \leq |S| \Rightarrow$ either $i - p \geq 1$ or $i + q \leq |S|$.

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf '65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

Proof. We show that $q - p$ is a period of S.



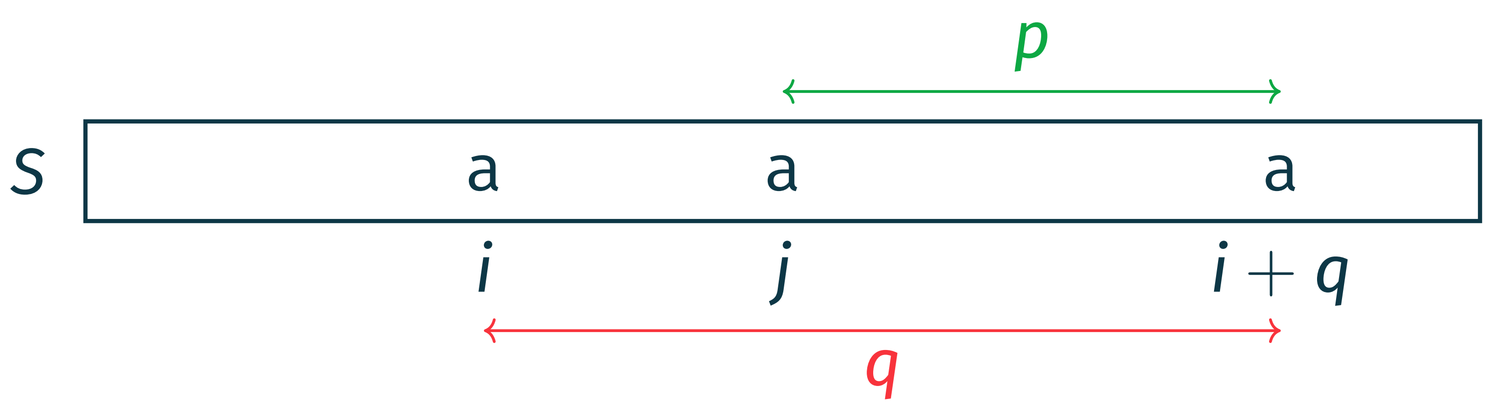
Let $j = i + q - p$.

$p + q \leq |S| \Rightarrow$ either $i - p \geq 1$ or $i + q \leq |S|$.

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf '65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

Proof. We show that $q - p$ is a period of S.



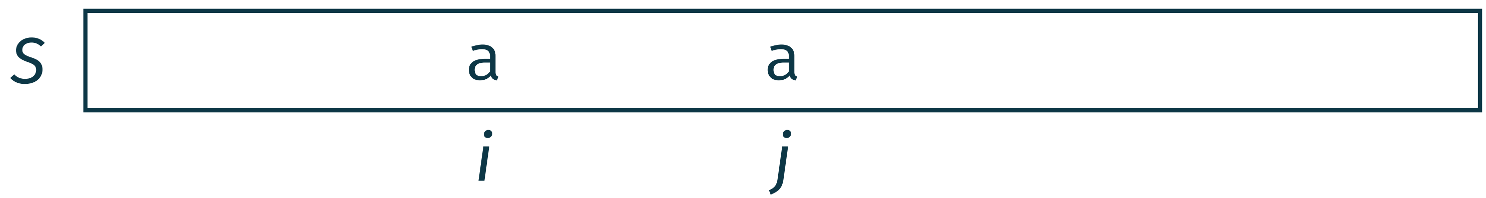
Let $j = i + q - p$.

$p + q \leq |S| \Rightarrow$ either $i - p \geq 1$ or $i + q \leq |S|$.

The periodicity lemma (weak version)

Periodicity Lemma (weak version) [Fine and Wilf ’65] If $p < q$ are periods of a string S and satisfy $p + q \leq |S|$, then $gcd(p, q)$ is also a period of S.

Proof. We show that $q - p$ is a period of S.



Let $j = i + q - p$.

$p + q \leq |S| \Rightarrow$ either $i - p \geq 1$ or $i + q \leq |S|$.

$(p, q) \rightarrow (p, q - p) \rightarrow \cdots \rightarrow (p, q \bmod p)$

This yields $gcd(p, q)$ as in Euclid’s algorithm (for computing the gcd of p and q).

Primitivity

A string is primitive if it is not the power of another string.

Primitivity

A string is primitive if it is not the power of another string.

Examples: ababa is primitive while ababab is not, as it is the 3rd power of ab.

Primitivity

A string is primitive if it is not the power of another string.

Examples: $\mathtt{ababa}$ is primitive while $\mathtt{ababab}$ is not, as it is the 3rd power of $\mathtt{ab}$.

$\mathtt{ab}$ is the primitive root of $\mathtt{ababab} = (\mathtt{ab})^3$.

Primitivity

A string is primitive if it is not the power of another string.

Examples: $\mathtt{ababa}$ is primitive while $\mathtt{ababab}$ is not, as it is the 3rd power of $\mathtt{ab}$.

$\mathtt{ab}$ is the primitive root of $\mathtt{ababab} = (\mathtt{ab})^3$.

Observation: For any string S, the prefix $S[\,1\,..\,\mathrm{per}(S)\,]$ is primitive.

Primitivity

A string is primitive if it is not the power of another string.

Examples: ababa is primitive while ababab is not, as it is the 3rd power of ab.

ab is the primitive root of $\mathtt{ababab} = (\mathtt{ab})^3$.

Observation: For any string S, the prefix $S[1..\mathrm{per}(S)]$ is primitive.

Primitivity

A string is primitive if it is not the power of another string.

Examples: $\mathtt{ababa}$ is primitive while $\mathtt{ababab}$ is not, as it is the 3rd power of $\mathtt{ab}$.

$\mathtt{ab}$ is the primitive root of $\mathtt{ababab} = (\mathtt{ab})^3$.

Observation: For any string S, the prefix $S[\,1\,..\,\mathrm{per}(S)\,]$ is primitive.

Primitivity Lemma: A primitive string U does not have any internal occurrence in UU. (In other words, U does not match any of its rotations.)

Primitivity

A string is primitive if it is not the power of another string.

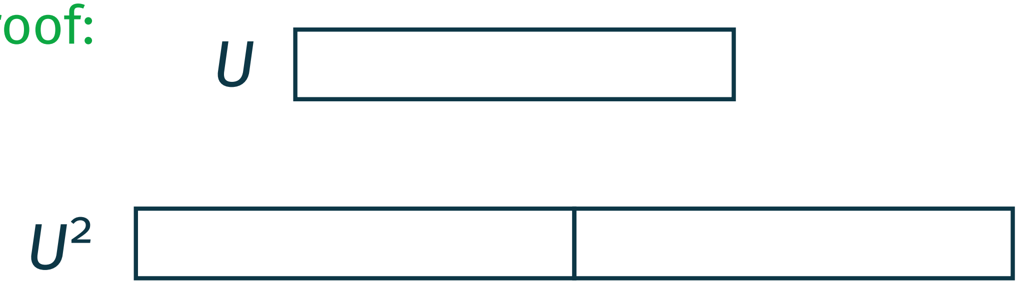
Examples: $\mathtt{ababa}$ is primitive while $\mathtt{ababab}$ is not, as it is the 3rd power of $\mathtt{ab}$.

$\mathtt{ab}$ is the primitive root of $\mathtt{ababab} = (\mathtt{ab})^3$.

Observation: For any string S, the prefix $S[1..\mathrm{per}(S)]$ is primitive.

Primitivity Lemma: A primitive string U does not have any internal occurrence in UU. (In other words, U does not match any of its rotations.)

Proof:



Primitivity

A string is primitive if it is not the power of another string.

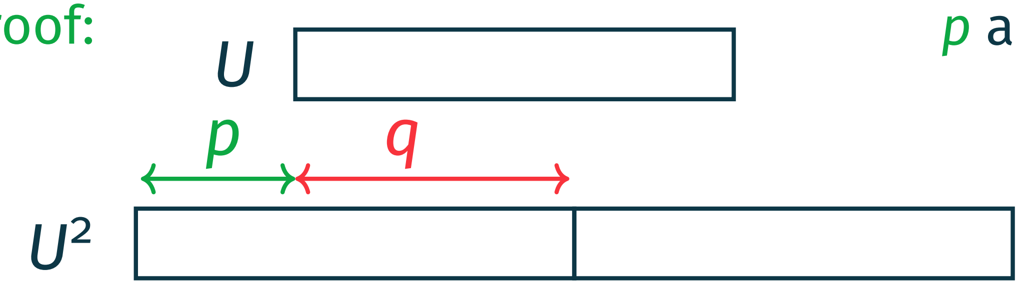
Examples: $\texttt{ababa}$ is primitive while $\texttt{ababab}$ is not, as it is the 3rd power of $\texttt{ab}$.

$\texttt{ab}$ is the primitive root of $\texttt{ababab} = (\texttt{ab})^3$.

Observation: For any string S, the prefix $S[1..\mathrm{per}(S)]$ is primitive.

Primitivity Lemma: A primitive string U does not have any internal occurrence in UU. (In other words, U does not match any of its rotations.)

Proof:



p and q are periods of U and $p + q = |U|$

Primitivity

A string is primitive if it is not the power of another string.

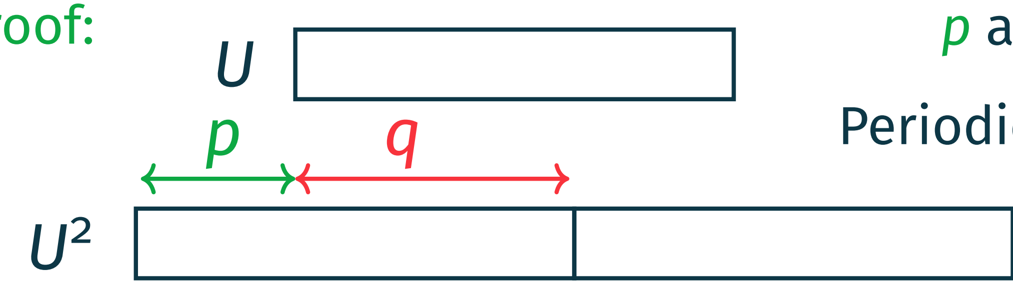
Examples: ababa is primitive while ababab is not, as it is the 3rd power of ab.

ab is the primitive root of ababab $= (\mathrm{ab})^3$.

Observation: For any string S, the prefix $S[\,1\,..\,\mathrm{per}(S)\,]$ is primitive.

Primitivity Lemma: A primitive string U does not have any internal occurrence in UU. (In other words, U does not match any of its rotations.)

Proof:



p and q are periods of U and $p + q = |U|$

Periodicity lemma $\Rightarrow gcd(p, q)$ a period of U!

Primitivity

A string is primitive if it is not the power of another string.

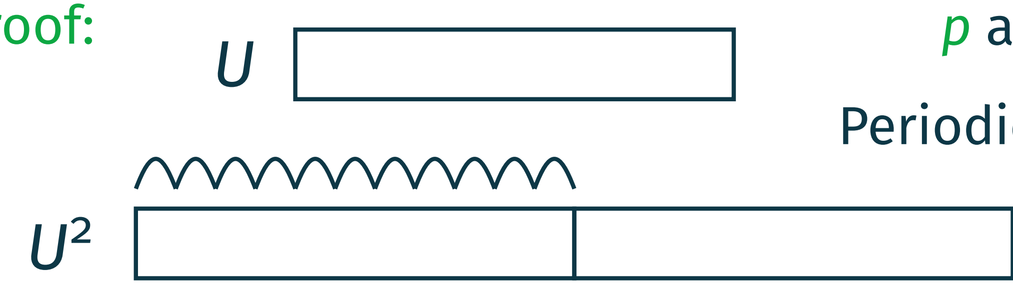
Examples: ababa is primitive while ababab is not, as it is the 3rd power of ab.

ab is the primitive root of ababab $= (\mathrm{ab})^3$.

Observation: For any string S, the prefix $S[\,1\,..\,\mathrm{per}(S)\,]$ is primitive.

Primitivity Lemma: A primitive string U does not have any internal occurrence in UU. (In other words, U does not match any of its rotations.)

Proof:



p and q are periods of U and $p + q = |U|$

Periodicity lemma $\Rightarrow gcd(p, q)$ a period of U!

U is a power of $U[\,1\,..\,gcd(p, q)\,]$

Algorithmic uses of periodicity

Algorithmic uses of periodicity

It is very often the case that an algorithmic problem on strings admits a simple efficient solution if there is no periodicity.

Algorithmic uses of periodicity

It is very often the case that an algorithmic problem on strings admits a simple efficient solution if there is no periodicity.

In the presence of periodicity, one might be able to use the extra structure to still obtain an efficient solution – but sometimes not/less simple!

Algorithmic uses of periodicity

It is very often the case that an algorithmic problem on strings admits a simple efficient solution if there is no periodicity.

In the presence of periodicity, one might be able to use the extra structure to still obtain an efficient solution – but sometimes not/less simple!

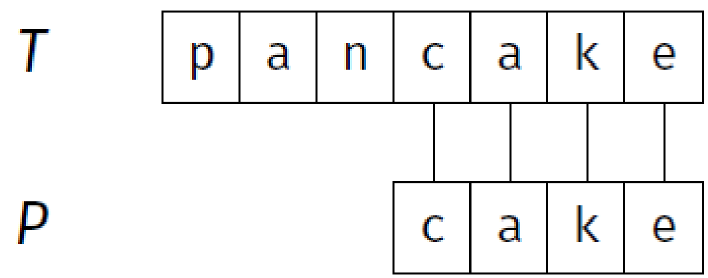
The solutions to many string algorithmic problems distinguish between the aperiodic and periodic cases.

A prime example: Pattern matching and the Morris-Pratt algorithm I

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.



A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T	a b a b a c a b c d e a b c a b a b a b a a b c b c c
P	a b a b a b a a b c b

Let us consider a naïve sliding window approach that takes $\mathcal{O}(nm)$ time.

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T	a b a b a c a b c d e a b c a b a b a b a a b c b c c
P	a b a b a b a a b c b

Let us consider a naïve sliding window approach that takes $\mathcal{O}(nm)$ time.

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T a b a b a c a b c d e a b c a b a b a b a a b c b c c

P a b a b a b a a b c b

Let us consider a naïve sliding window approach that takes $\mathcal{O}(nm)$ time.

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T $\boxed{\mathtt{a\,b\,a\,b\,a\,c\,a\,b\,c\,d\,e\,a\,b\,c\,a\,b\,a\,b\,a\,b\,a\,a\,b\,c\,b\,c\,c}}$

What if there is no periodicity whatsoever in the pattern?

(That is, what if none of P's substrings have a non-trivial period?)

For example, what if $P = \mathtt{a\,b\,c\,d\,e\,f\,g\,h\,i\,j\,k}$?

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T `a b a b a c a b c d e a b c a b a b a b a a b c b c c`

P `a b c d e f g h i j k`

What if there is no periodicity whatsoever in the pattern?

(That is, what if none of P's substrings have a non-trivial period?)

For example, what if $P = \mathtt{a\,b\,c\,d\,e\,f\,g\,h\,i\,j\,k}$?

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T `a b a b a c a b c d e a b c a b a b a b a a b c b c c`

P `a b c d e f g h i j k`

What if there is no periodicity whatsoever in the pattern?

(That is, what if none of P's substrings have a non-trivial period?)

For example, what if $P =$ `a b c d e f g h i j k`?

We compare each letter of T with at most two letters of P. $\rightarrow \mathcal{O}(n)$ time!

A prime example: Pattern matching and the Morris-Pratt algorithm I

Pattern Matching

Given a text T of length n and a pattern P of length m, compute the occurrences of P in T.

T `a b a b a c a b c d e a b c a b a b a b a a b c b c c`

P `a b c d e f g h i j k`

What if there is no periodicity whatsoever in the pattern?

(That is, what if none of P's substrings have a non-trivial period?)

For example, what if $P =$ `a b c d e f g h i j k`?

We compare each letter of T with at most two letters of P. $\rightarrow \mathcal{O}(n)$ time!

A prime example: Pattern matching and the Morris-Pratt algorithm II

A prime example: Pattern matching and the Morris-Pratt algorithm II

Morris-Pratt: A sliding window algorithm with the following rule for shifting:

shift by the period of the partial (or full) match!

A prime example: Pattern matching and the Morris-Pratt algorithm II

Morris-Pratt: A sliding window algorithm with the following rule for shifting: shift by the period of the partial (or full) match!

T	a b a b a c a b c d e a b c a b a b a b a a b c b c c
P	a b a b a b a a b c b

A prime example: Pattern matching and the Morris-Pratt algorithm II

Morris-Pratt: A sliding window algorithm with the following rule for shifting:

shift by the period of the partial (or full) match!

T a b a b a c a b c d e a b c a b a b a b a a b c b c c

P a b a b a b a a b c b

P a b a b a b a a b c b

We do not lose any occurrences, as any smaller shift would give a mismatch.

A prime example: Pattern matching and the Morris-Pratt algorithm II

Morris-Pratt: A sliding window algorithm with the following rule for shifting: shift by the period of the partial (or full) match!

T	a b a b a c a b c d e a b c a b a b a b a a b c b c c
P	a b a b a b a a b c b
P	a b a b a b a a b c b

We do not lose any occurrences, as any smaller shift would give a mismatch.

We do not need to perform any comparisons involving letters of T that were already matched!

A prime example: Pattern matching and the Morris-Pratt algorithm II

Morris-Pratt: A sliding window algorithm with the following rule for shifting: shift by the period of the partial (or full) match!

T a b a b a c a b c d e a b c a b a b a b a a b c b c c

P a b a b a b a a b c b

P a b a b a b a a b c b

We do not lose any occurrences, as any smaller shift would give a mismatch.

We do not need to perform any comparisons involving letters of T that were already matched!

Each successful letter comparison consumes a letter of T, while each unsuccessful letter comparison shifts P. $\rightarrow \mathcal{O}(n)$ time!

The Structure of Exact Pattern Matching

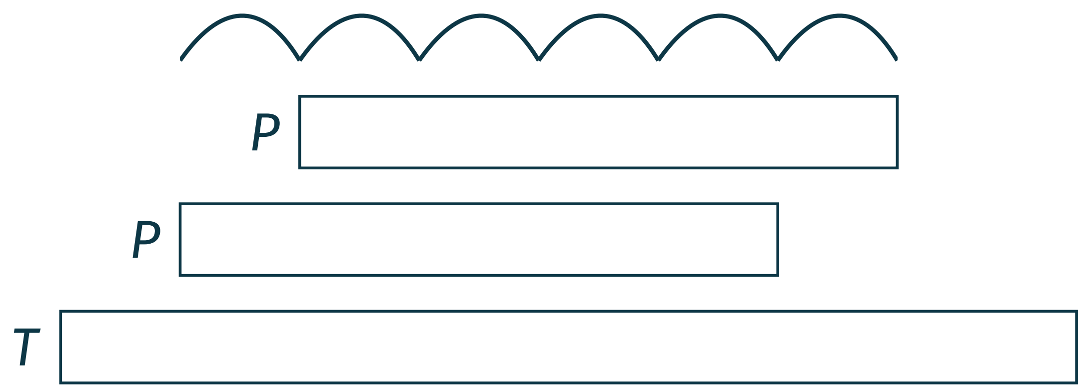
Fact [folklore] Given a pattern P of length m and a text T of length $n \leq 3/2\, m$ at least one of the following holds:

Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$ at least one of the following holds:

- The pattern P has at most one occurrence in T.

Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^3\!/_2\, m$ at least one of the following holds:

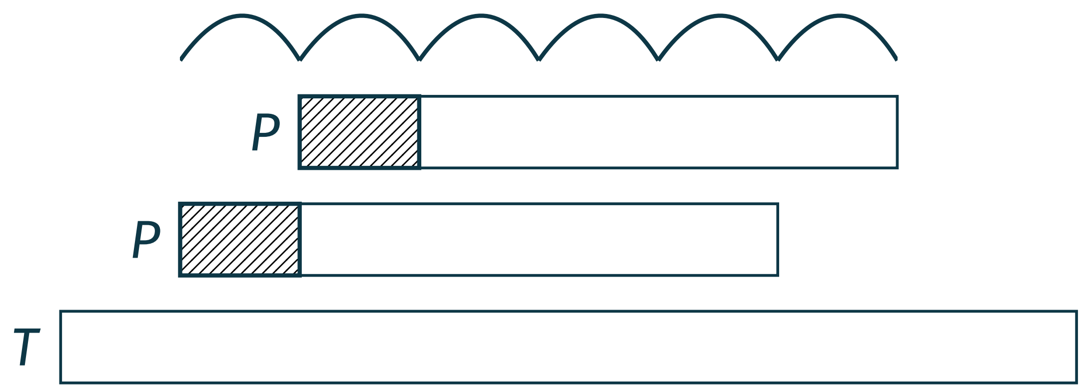
- The pattern P has at most one occurrence in T.
- The pattern P is periodic.



The Structure of Exact Pattern Matching

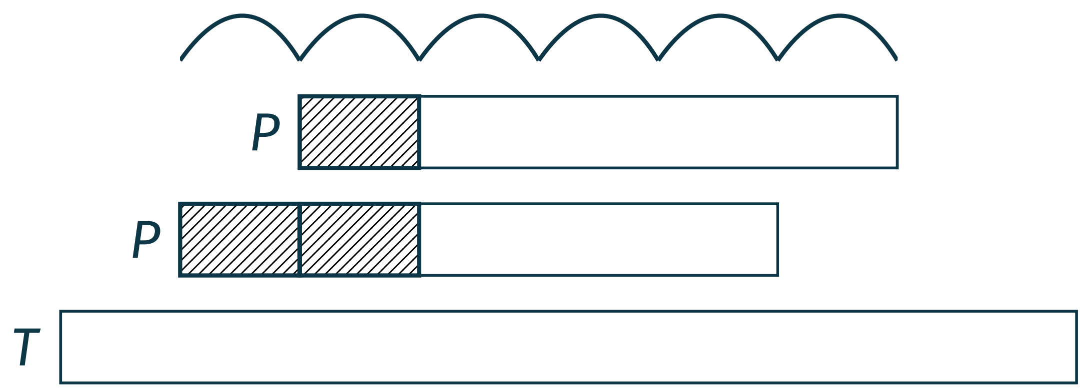
Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$ at least one of the following holds:

- The pattern P has at most one occurrence in T.
- The pattern P is periodic.



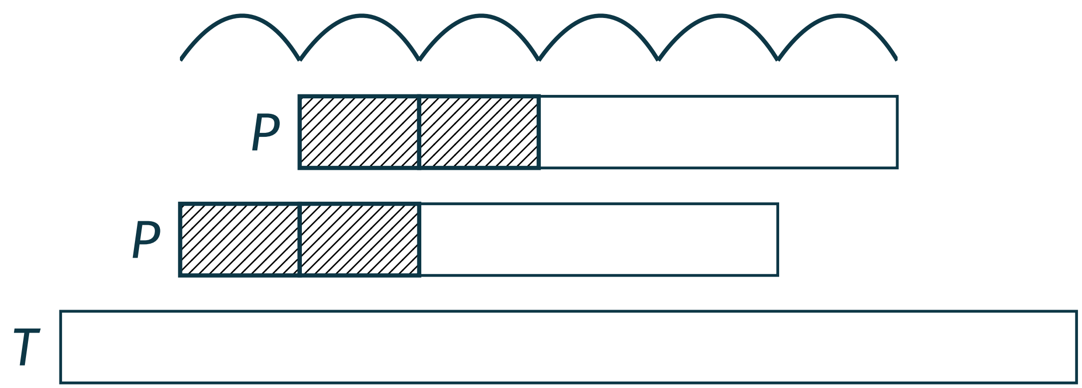
Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$ at least one of the following holds:

- The pattern P has at most one occurrence in T.
- The pattern P is periodic.



Fact [folklore] Given a pattern P of length m and a text T of length $n \leq 3/2\, m$ at least one of the following holds:

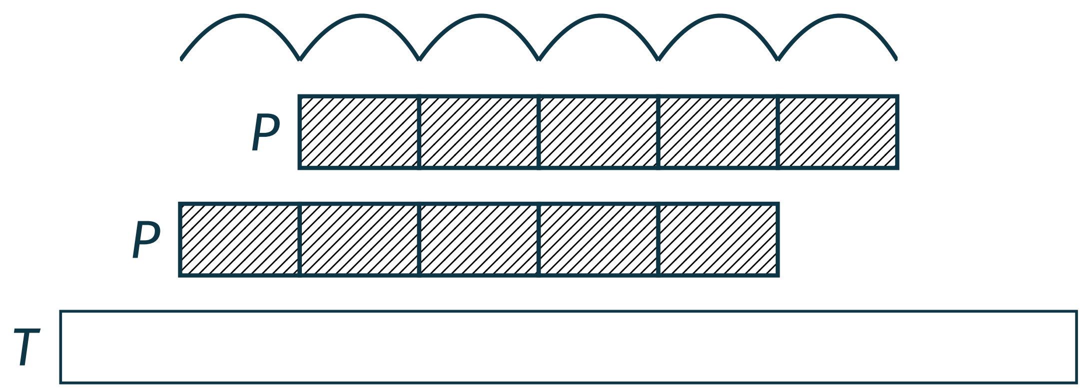
- The pattern P has at most one occurrence in T.
- The pattern P is periodic.



The Structure of Exact Pattern Matching

Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^3\!/_2\, m$ at least one of the following holds:

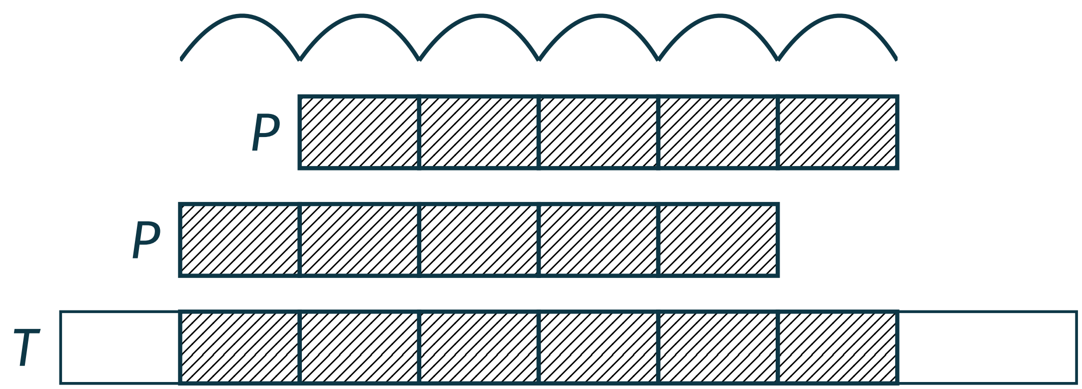
- The pattern P has at most one occurrence in T.
- The pattern P is periodic.



The Structure of Exact Pattern Matching

Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$ at least one of the following holds:

- The pattern P has at most one occurrence in T.
- The pattern P is periodic.

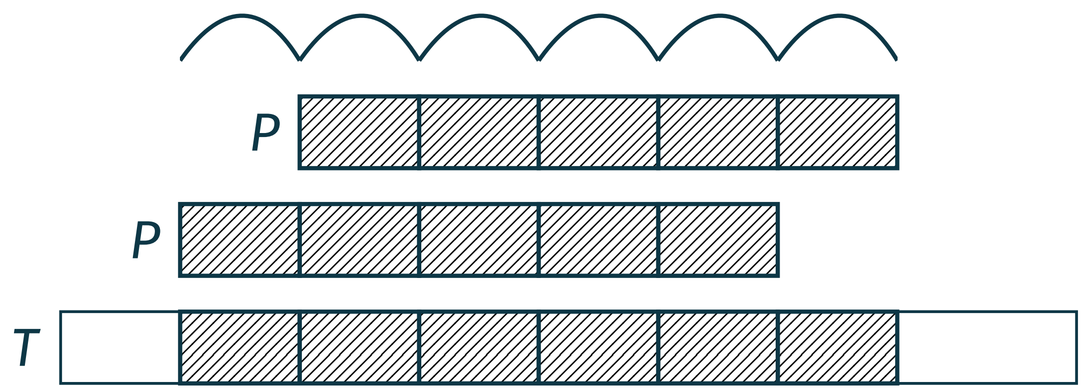


The fragment of T spanned by P's occurrences is periodic as well.

The Structure of Exact Pattern Matching

Fact [folklore] Given a pattern P of length m and a text T of length $n \leq {}^3\!/_2\, m$ at least one of the following holds:

- The pattern P has at most one occurrence in T.
- The pattern P is periodic.



The standard trick: Our assumption on the length of the text is not restrictive. If the text is much longer that the pattern, we can always consider separately $\mathcal{O}(n/m)$ fragments of T of length $\leq {}^3\!/_2\, m$ that overlap by $m - 1$ positions.

Exact pattern matching can be very restrictive

Exact pattern matching can be very restrictive

Think of human spelling mistakes or DNA sequencing errors, for example.

Exact pattern matching can be very restrictive

Think of human spelling mistakes or DNA sequencing errors, for example.

In *approximate pattern matching*, we are interested in finding substrings of T that are *similar* to P. Today, we will talk about the two most commonly encountered metrics in this context: the Hamming distance and the edit distance.

Exact pattern matching can be very restrictive

Think of human spelling mistakes or DNA sequencing errors, for example.

In *approximate pattern matching*, we are interested in finding substrings of T that are *similar* to P. Today, we will talk about the two most commonly encountered metrics in this context: the Hamming distance and the edit distance.

The Hamming distance of two equal-length strings is the number of positions on which they differ (equivalently, the number of mismatches).

Exact pattern matching can be very restrictive

Think of human spelling mistakes or DNA sequencing errors, for example.

In *approximate pattern matching*, we are interested in finding substrings of T that are *similar* to P. Today, we will talk about the two most commonly encountered metrics in this context: the Hamming distance and the edit distance.

The Hamming distance of two equal-length strings is the number of positions on which they differ (equivalently, the number of mismatches).

The edit distance of two strings is the minimum number of edits (letter insertions, deletions, substitutions) required to transform one string into the other.

Exact pattern matching can be very restrictive

Think of human spelling mistakes or DNA sequencing errors, for example.

In *approximate pattern matching*, we are interested in finding substrings of T that are *similar* to P. Today, we will talk about the two most commonly encountered metrics in this context: the Hamming distance and the edit distance.

The Hamming distance of two equal-length strings is the number of positions on which they differ (equivalently, the number of mismatches).

The edit distance of two strings is the minimum number of edits (letter insertions, deletions, substitutions) required to transform one string into the other.

We will discuss the structure of approximate pattern matching under each of these metrics and see how the structural analysis yields efficient algorithms in several settings.

Approximate Pattern Matching under the Hamming Distance and the Edit Distance

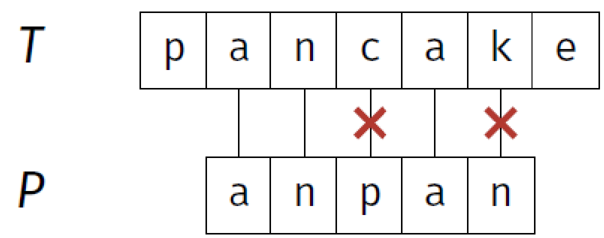
Approximate Pattern Matching under the Hamming Distance and the Edit Distance

Pattern Matching under Hamming Distance

Given a text T, a pattern P, and an integer threshold k, compute the length-$|P|$ substrings of T that are at Hamming distance at most k from P.

Pattern Matching under Hamming Distance

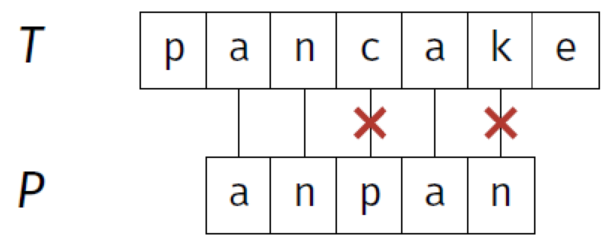
Given a text T, a pattern P, and an integer threshold k, compute the length-$|P|$ substrings of T that are at Hamming distance at most k from P.



Approximate Pattern Matching under the Hamming Distance and the Edit Distance

Pattern Matching under Hamming Distance

Given a text T, a pattern P, and an integer threshold k, compute the length-$|P|$ substrings of T that are at Hamming distance at most k from P.



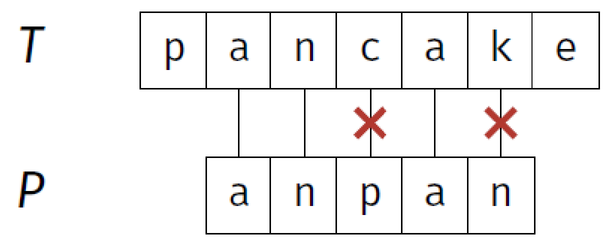
Pattern Matching under Edit Distance

Given a text T, a pattern P, and an integer threshold k, compute the (starting positions of) substrings of T that are at edit distance at most k from P.

Approximate Pattern Matching under the Hamming Distance and the Edit Distance

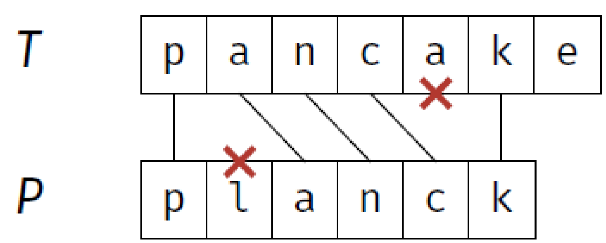
Pattern Matching under Hamming Distance

Given a text T, a pattern P, and an integer threshold k, compute the length-$|P|$ substrings of T that are at Hamming distance at most k from P.



Pattern Matching under Edit Distance

Given a text T, a pattern P, and an integer threshold k, compute the (starting positions of) substrings of T that are at edit distance at most k from P.



The structure was understood only recently!

The structure was understood only recently!

There is a long history of algorithmic results for these problems, which we will discuss later.

The structure was understood only recently!

There is a long history of algorithmic results for these problems, which we will discuss later.

Some of these algorithms heavily relied on exploiting the periodic structure of P and T, and even implied some (somewhat weak) structural results.

The structure was understood only recently!

There is a long history of algorithmic results for these problems, which we will discuss later.

Some of these algorithms heavily relied on exploiting the periodic structure of P and T, and even implied some (somewhat weak) structural results.

The first explicitly stated structural result (for Hamming distance) was proved by Bringmann, Künnemann, and Wellnitz in 2019.

The structure was understood only recently!

There is a long history of algorithmic results for these problems, which we will discuss later.

Some of these algorithms heavily relied on exploiting the periodic structure of P and T, and even implied some (somewhat weak) structural results.

The first explicitly stated structural result (for Hamming distance) was proved by Bringmann, Künnemann, and Wellnitz in 2019.

This result was tightened and extended to also cover approximate pattern matching under the edit distance by C., Kociumaka, and Wellnitz in 2020.

Partitioning the pattern and extending seeds

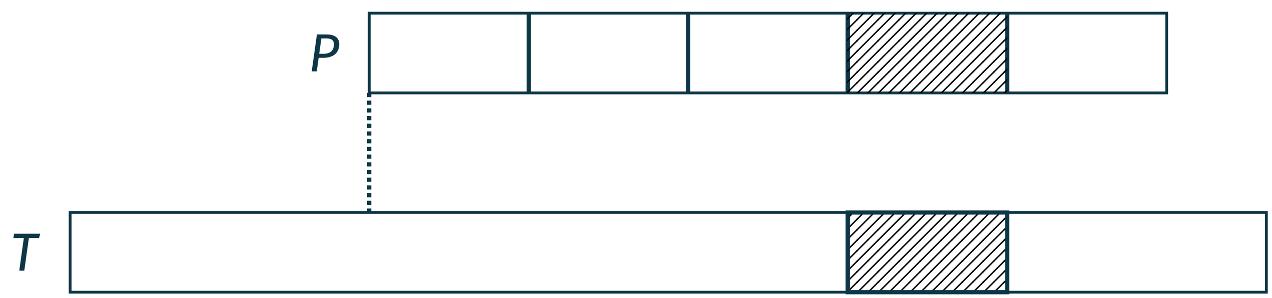
Partitioning the pattern and extending seeds

We will now see a technique that is also useful in practice.

Partitioning the pattern and extending seeds

We will now see a technique that is also useful in practice.

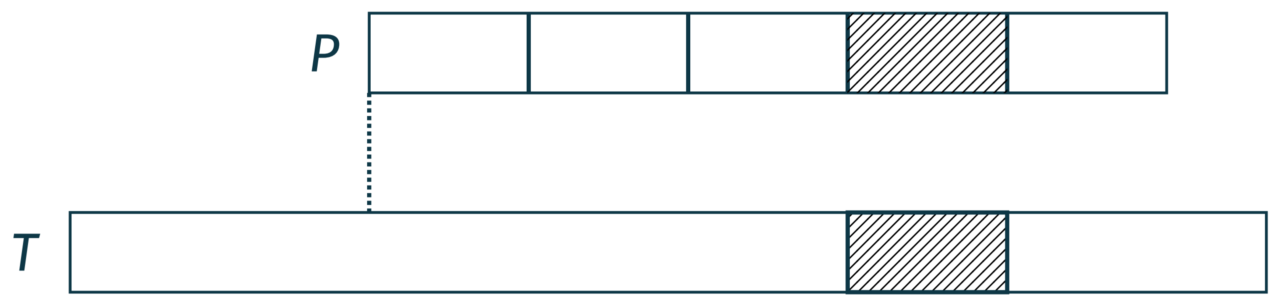
Observation: Let us partition P into $k+1$ (roughly) equal chunks, each of length $\approx m/(k+1)$. In any approximate match of P in T, at least one of the chunks must be matched exactly.



Partitioning the pattern and extending seeds

We will now see a technique that is also useful in practice.

Observation: Let us partition P into $k+1$ (roughly) equal chunks, each of length $\approx m/(k+1)$. In any approximate match of P in T, at least one of the chunks must be matched exactly.

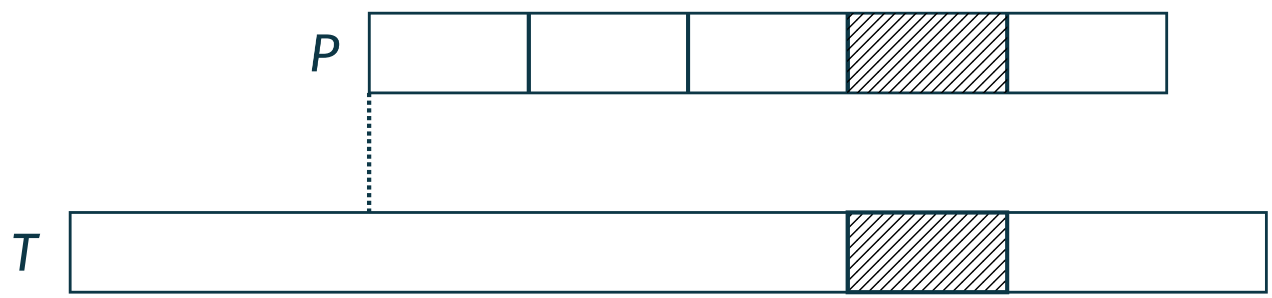


Algorithm strategy: Find the exact matches of each chunk in T (called seeds) and try to extend them to approximate matches of P.

Partitioning the pattern and extending seeds

We will now see a technique that is also useful in practice.

Observation: Let us partition P into $k+1$ (roughly) equal chunks, each of length $\approx m/(k+1)$. In any approximate match of P in T, at least one of the chunks must be matched exactly.



Algorithm strategy: Find the exact matches of each chunk in T (called seeds) and try to extend them to approximate matches of P.

Observation: If a chunk P_i is aperiodic, its occurrences cannot overlap by more than $|P_i|/2$ positions $\Rightarrow$ at most $n/(|P_i|/2)$ occurrences, which is $\mathcal{O}(k \cdot n/m)$.

The Marking Trick (for Hamming distance and $n \leq 2m$)

The Marking Trick (for Hamming distance and $n \leq 2m$)

Complexity of the seeding technique in an aperiodic case:

- $\mathcal{O}(k)$ calls to exact pattern matching, one for each chunk;
- $\mathcal{O}(k^2)$ attempts to extend a seed ($\mathcal{O}(k)$ for each of the $\mathcal{O}(k)$ chunks).

  Each seed gives a candidate starting position for an approximate occurrence.

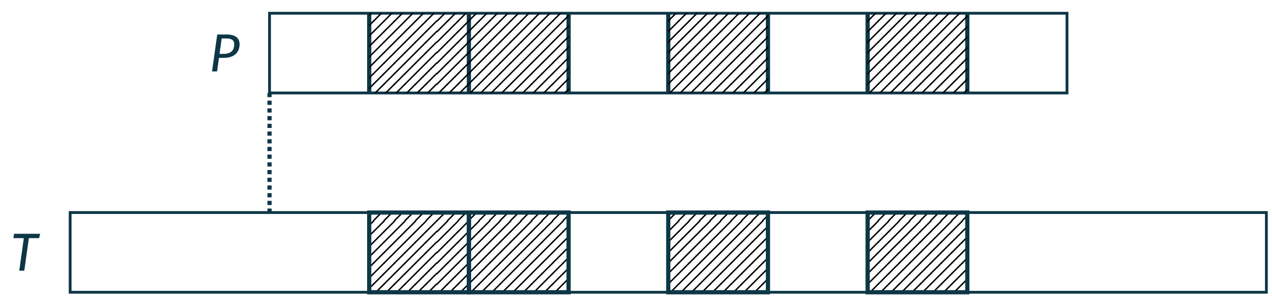
  This gives us an $\mathcal{O}(k^2)$ bound on approximate occurrences in the case where all chunks are aperiodic!

The Marking Trick (for Hamming distance and $n \leq 2m$)

Complexity of the seeding technique in an aperiodic case:

- $\mathcal{O}(k)$ calls to exact pattern matching, one for each chunk;
- $\mathcal{O}(k^2)$ attempts to extend a seed ($\mathcal{O}(k)$ for each of the $\mathcal{O}(k)$ chunks).

Marking trick: Partition P into $2k$ chunks, each of length $\approx m/(2k)$. In any approximate match of P in T, at least k of the chunks must be matched exactly.

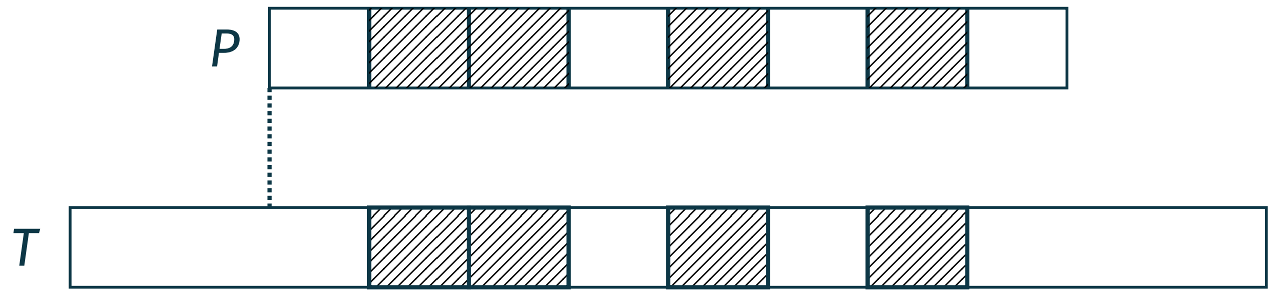


The Marking Trick (for Hamming distance and $n \leq 2m$)

Complexity of the seeding technique in an aperiodic case:

- $\mathcal{O}(k)$ calls to exact pattern matching, one for each chunk;
- $\mathcal{O}(k^2)$ attempts to extend a seed ($\mathcal{O}(k)$ for each of the $\mathcal{O}(k)$ chunks).

Marking trick: Partition P into $2k$ chunks, each of length $\approx m/(2k)$. In any approximate match of P in T, at least k of the chunks must be matched exactly.



For each exact match of a chunk, give a mark to the candidate starting position of an approximate occurrence in T. Then, we only need to verify candidate starting positions with $\geq k$ marks. These are $\mathcal{O}(k)$ as we have $\mathcal{O}(k^2)$ marks overall.

The First Structural Result for Hamming Distance [BKW'19]

Theorem (Bringmann-Künnemann-Wellnitz, SODA 2019)

Given a pattern P of length m, a text T of length $n \leq 2m$, and a threshold $k \leq m$, at least one of the following holds:

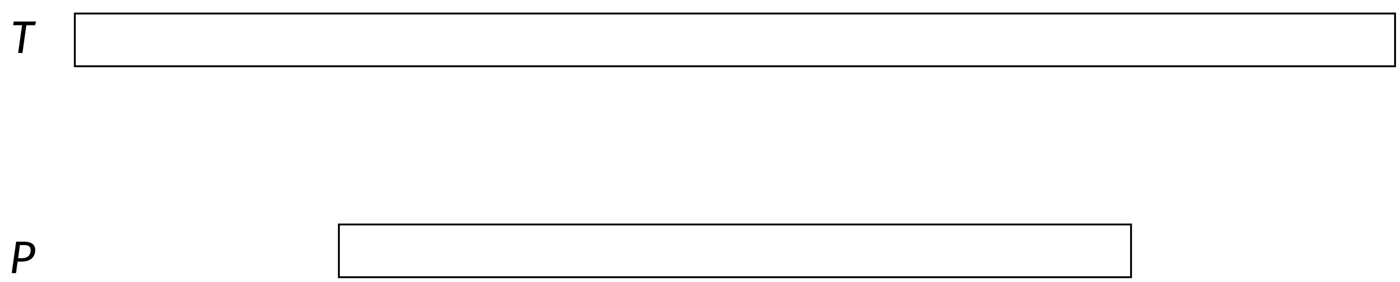
- The number of k-mismatch occurrences of P in T is $O(k^2)$.
- The pattern P is almost periodic (at HD $\leq 6k$ to a string with period $O(m/k)$).

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\le 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\le 6k$ to a string with period $O(m/k)$.

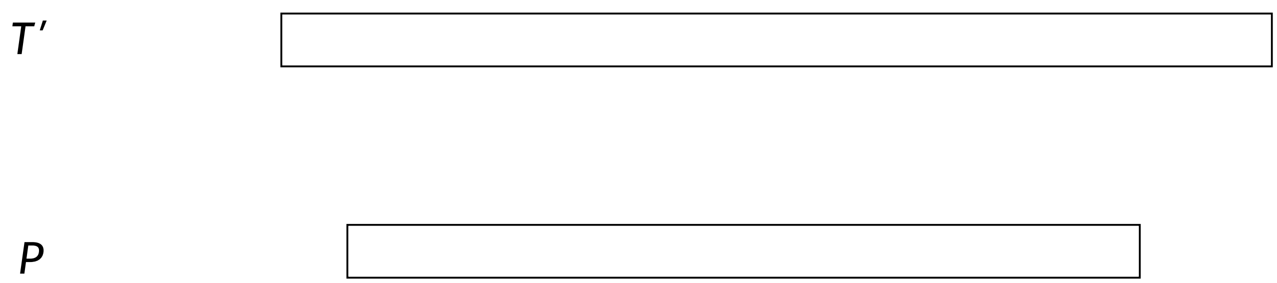


Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



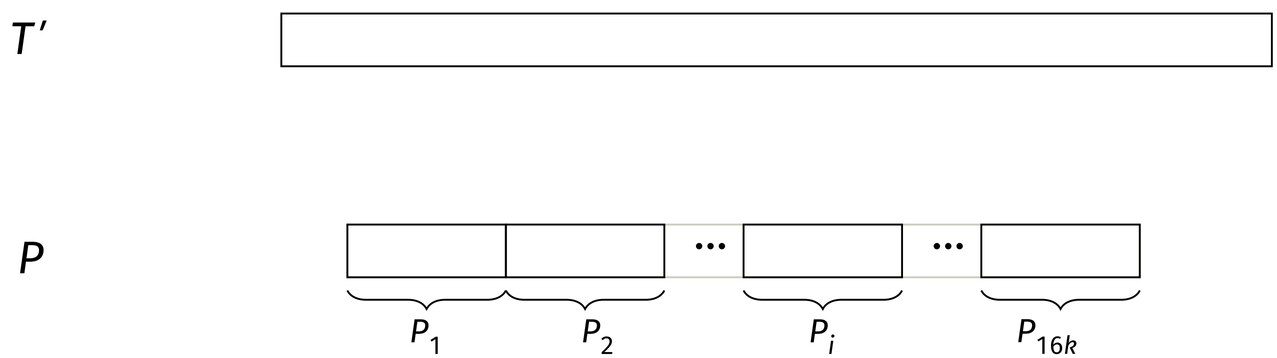
- Consider T': shortest substring of T that contains all k-mismatch occurrences of P.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



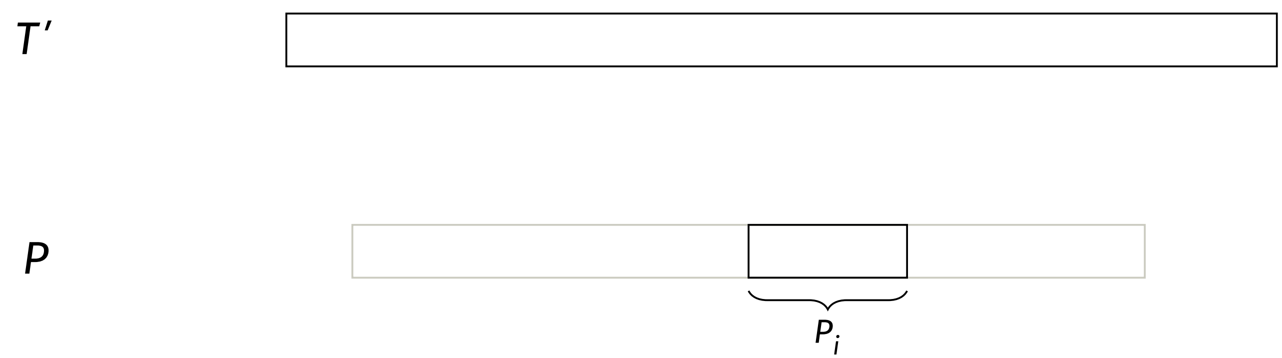
- Partition P into $16k$ parts P_i of (roughly) equal length.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



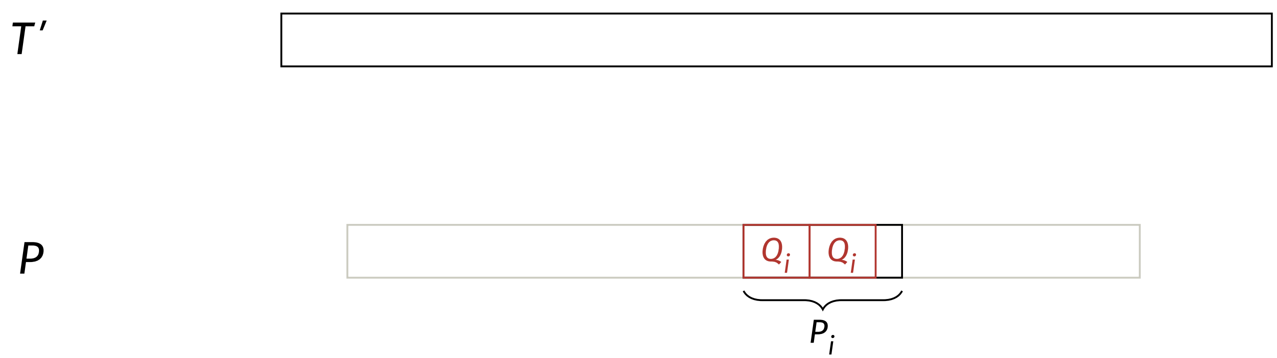
- In any k-mismatch occurrence of P in T' at least one of the P_i's must be matched **exactly**. Fix some P_i and assume that it is periodic; otherwise it only has $O(k)$ occurrences in T.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



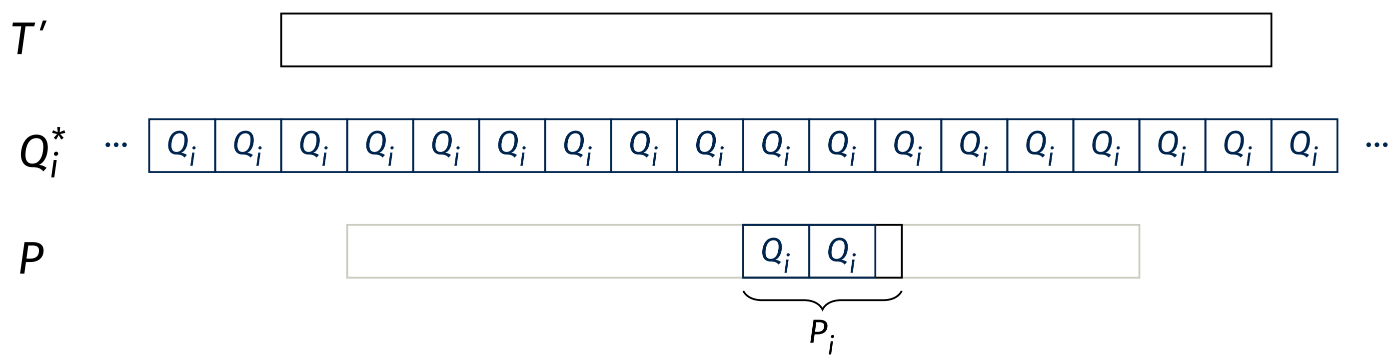
- Let Q_i be the prefix of P_i whose length is equal to the period of P_i.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\le 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\le 6k$ to a string with period $O(m/k)$.



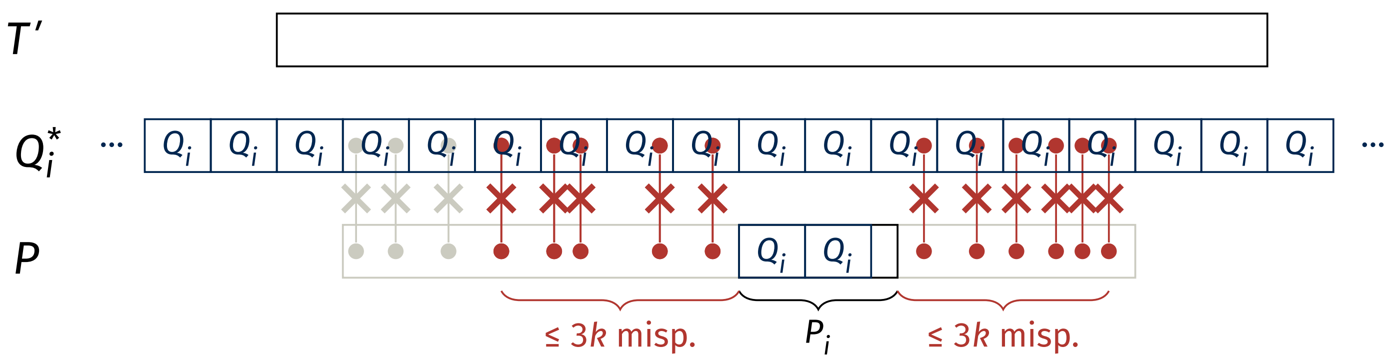
- Find the first $3k$ mismatches between P and Q_i^* before and after P_i.
  (We call such mismatches misperiods from now on.)

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



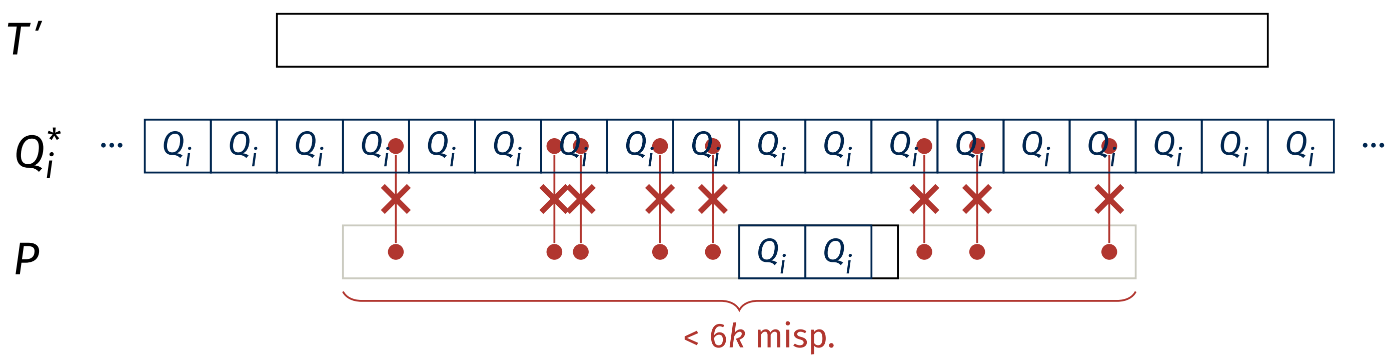
- Find the first $3k$ mismatches between P and Q_i^* before and after P_i.
(We call such mismatches misperiods from now on.)

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- **P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.**

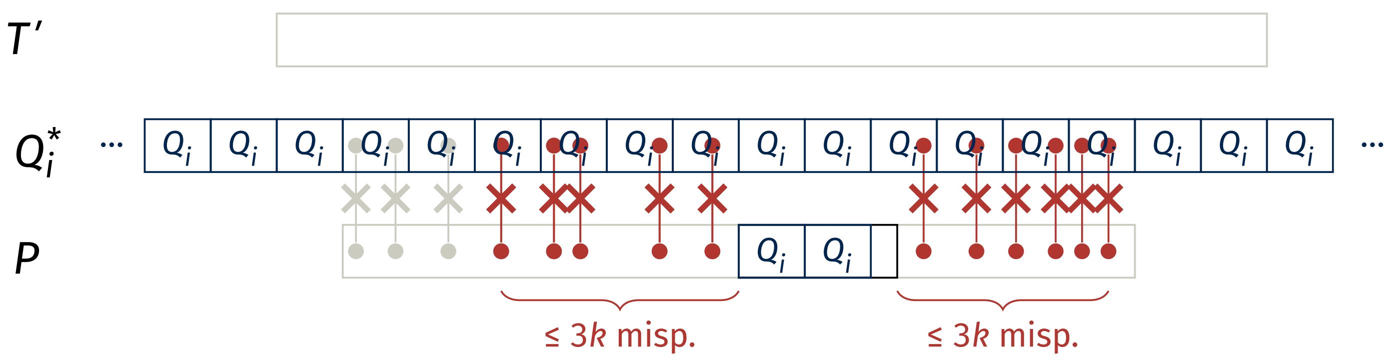


Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



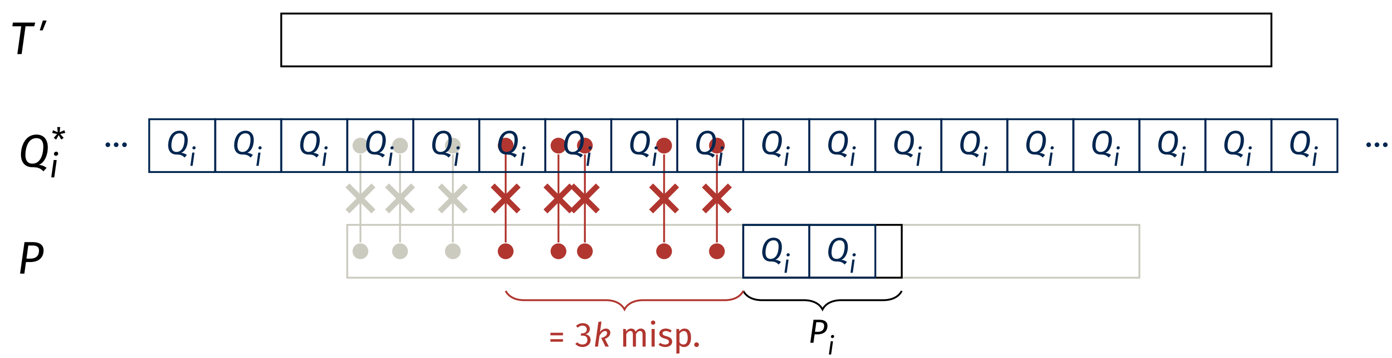
- We henceforth assume that we have found $3k$ misperiods in at least one of the two sides of P_i.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\le 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\le 6k$ to a string with period $O(m/k)$.



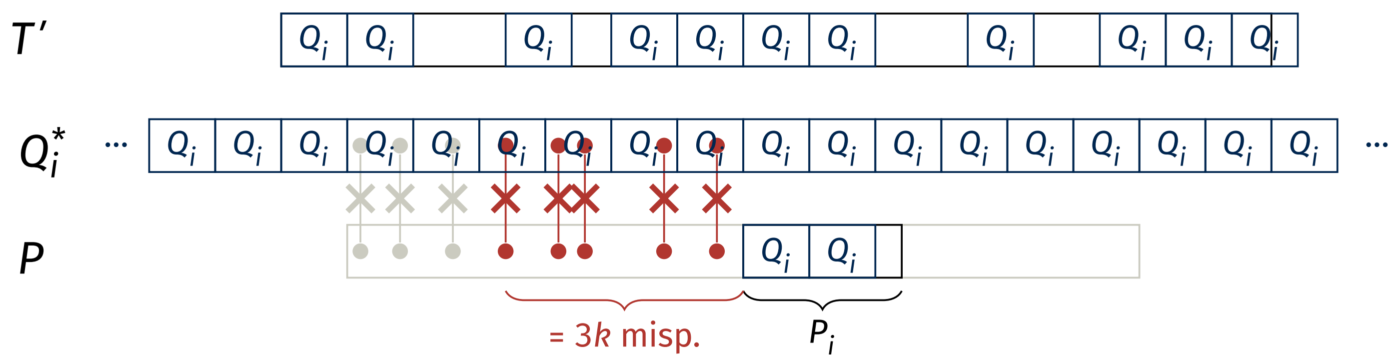
- We henceforth assume that we have found $3k$ misperiods in at least one of the two sides of P_i.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



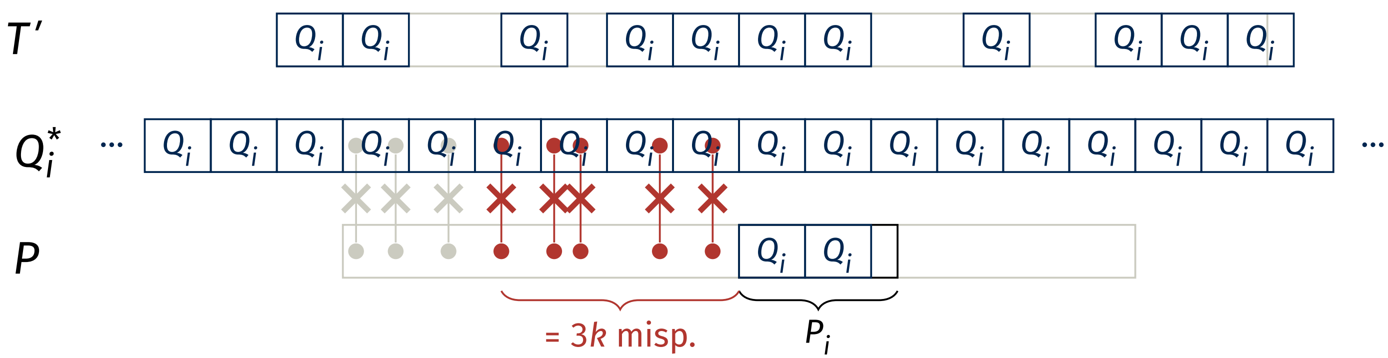
- Consider the occurrences of Q_i in T'.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



Problem

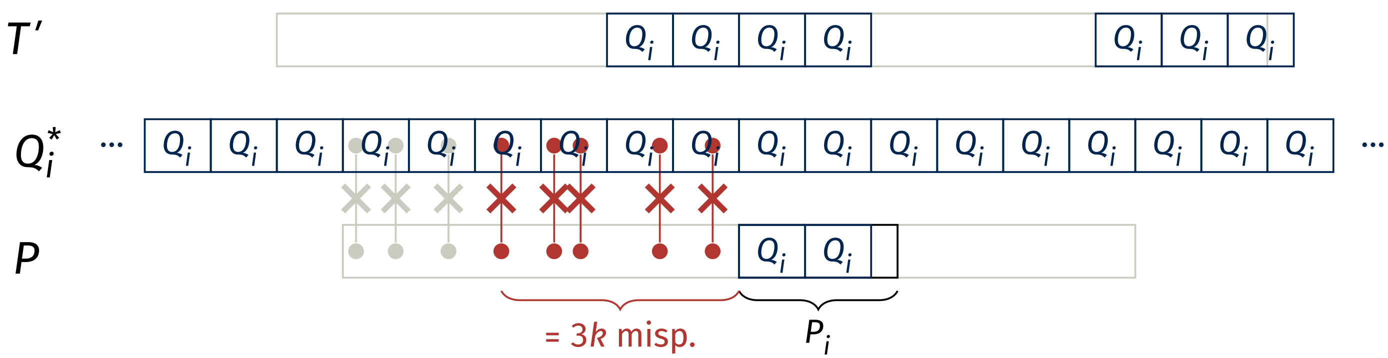
Up to $O(m)$ exact matches of Q_i in T'.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



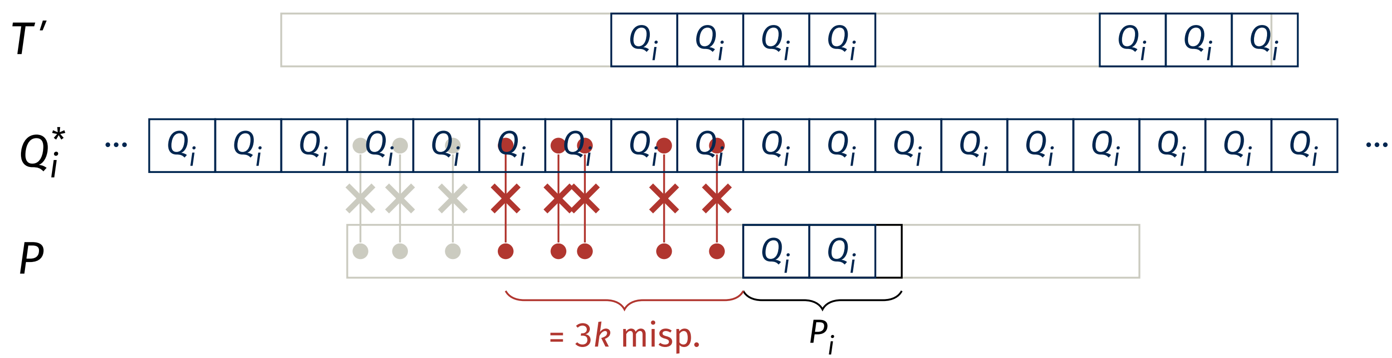
- Consider **power stretches** of Q_i in T' of length $\geq |P_i|$

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



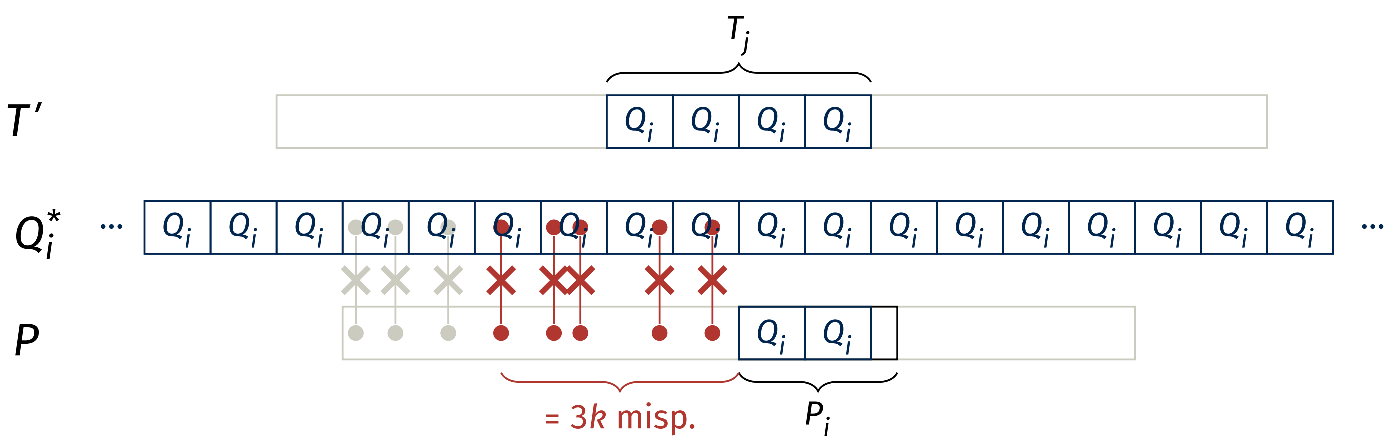
- Consider **power stretches** of Q_i in T' of length $\geq |P_i|$
  $\rightsquigarrow$ at most $150k$ different power stretches.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



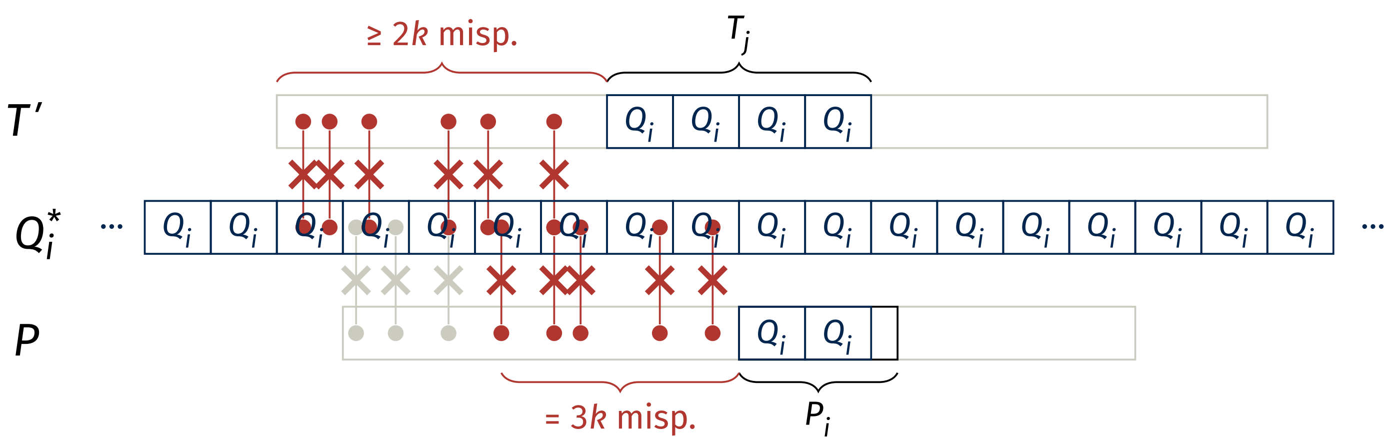
- Fix a power stretch T_j of Q_i in T'.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\le 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\le 6k$ to a string with period $O(m/k)$.



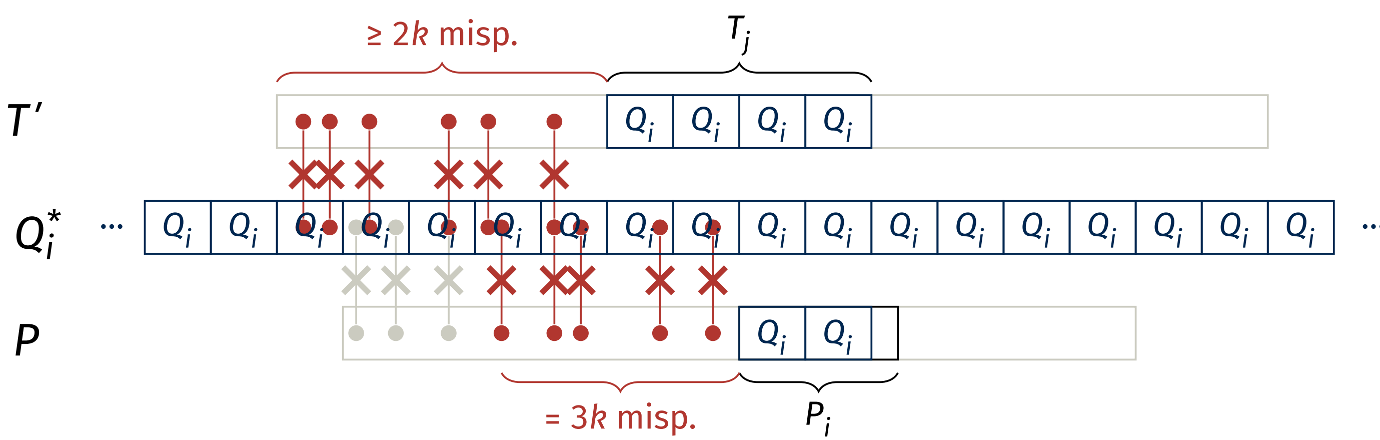
- Fix a power stretch T_j of Q_i in T'.

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is at most $1000k^2$.
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



Insight

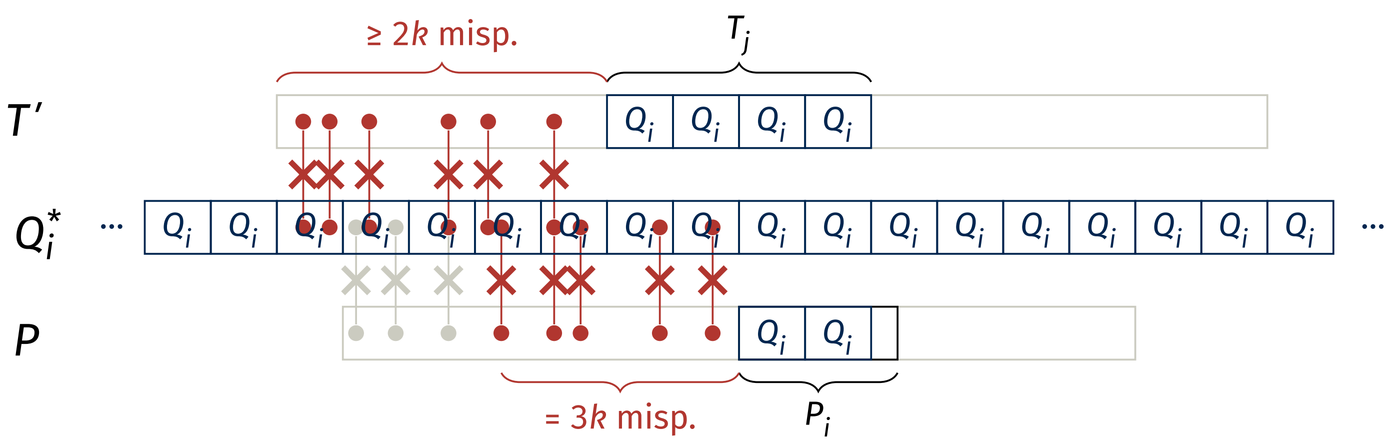
Must align at least one misperiod. (For intuition, consider the case where $T' = Q_i^t$ for some integer t. Then, for any exact match of P_i all misperiods in P yield mismatches between P and T'!)

Proof Overview

Structural Theorem (HD) [Bringmann-Künnemann-Wellnitz, SODA 2019]

For a pattern P of length m and a text T of length $\leq 2|P|$, at least one of the following holds:

- **The number of k-mismatch occurrences of P in T is at most $1000k^2$.**
- P is at Hamming distance $\leq 6k$ to a string with period $O(m/k)$.



Almost there

At most $O(k^4)$ k-mismatch occurrences: $O(k)$ choices for P_i, $O(k)$ choices for a power stretch, $O(k^2)$ pairs of aligned misperiods per combination.

Summary of the proof and a challenge: $O(k^2)$ bound using the marking trick?

- Partition the pattern into $16k$ chunks P_i of length $\approx m/16k$.
- Each aperiodic chunk has $O(k)$ exact occurrences. Each such occurrence gives a single candidate starting position for a k-mismatch occurrence of P. $O(k^2)$ overall.
- For each periodic chunk P_i, extend the periodicity in both sides, allowing $3k$ misp.
  - If all of P gets covered, we conclude that the pattern is almost periodic.
  - Else, for each of the $O(k)$ power stretches with period $\mathrm{per}(P_i)$ that contain occurrences of P_i, extend the periodicity in both sides, allowing $2k$ misperiods.
    Crucial observation: in any k-error occurrence that matches P_i exactly, a misperiod in P must be aligned with a misperiod in T.

Summary of the proof and a challenge: $O(k^2)$ bound using the marking trick?

- Partition the pattern into $16k$ chunks P_i of length $\approx m/16k$.
- Each aperiodic chunk has $O(k)$ exact occurrences. Each such occurrence gives a single candidate starting position for a k-mismatch occurrence of P. $O(k^2)$ overall.
- For each periodic chunk P_i, extend the periodicity in both sides, allowing $3k$ misp.
  - If all of P gets covered, we conclude that the pattern is almost periodic.
  - Else, for each of the $O(k)$ power stretches with period $\mathrm{per}(P_i)$ that contain occurrences of P_i, extend the periodicity in both sides, allowing $2k$ misperiods.
    Crucial observation: in any k-error occurrence that matches P_i exactly, a misperiod in P must be aligned with a misperiod in T.

Almost there

We have $O(k^4)$ k-mismatch occurrences: $O(k)$ choices for P_i, $O(k)$ choices for a power stretch, $O(k^2)$ pairs of aligned misp. per combination.

Summary of the proof and a challenge: $O(k^2)$ bound using the marking trick?

- Partition the pattern into $16k$ chunks P_i of length $\approx m/16k$.
- Each aperiodic chunk has $O(k)$ exact occurrences. Each such occurrence gives a single candidate starting position for a k-mismatch occurrence of P. $O(k^2)$ overall.
- For each periodic chunk P_i, extend the periodicity in both sides, allowing $3k$ misp.
  - If all of P gets covered, we conclude that the pattern is almost periodic.
  - Else, for each of the $O(k)$ power stretches with period $\text{per}(P_i)$ that contain occurrences of P_i, extend the periodicity in both sides, allowing $2k$ misperiods.
    Crucial observation: in any k-error occurrence that matches P_i exactly, k misperiods in P must be aligned with misperiods in T.

Nearly there

We have $O(k^3)$ k-mismatch occurrences: $O(k)$ choices for P_i, $O(k)$ choices for a power stretch, $O(k^2)$ pairs of aligned misp. per combination. Need $\geq k$ pairs of alinged misp.

Summary of the proof and a challenge: $O(k^2)$ bound using the marking trick?

- Partition the pattern into $16k$ chunks P_i of length $\approx m/16k$.
- Each aperiodic chunk has $O(k)$ exact occurrences. Each such occurrence gives a single candidate starting position for a k-mismatch occurrence of P. $O(k^2)$ overall.
- For each periodic chunk P_i, extend the periodicity in both sides, allowing $3k$ misp.
  - If all of P gets covered, we conclude that the pattern is almost periodic.
  - Else, for each of the $O(k)$ power stretches with period per(P_i) that contain occurrences of P_i, extend the periodicity in both sides, allowing $2k$ misperiods.
    Crucial observation: in any k-error occurrence that matches P_i exactly, k misperiods in P must be aligned with misperiods in T.
    At least k P_j's must nominate any potential starting position.

Finally there, maybe

At most $O(k^2)$ k-mismatch occurrences: $O(k)$ choices for P_i, $O(k)$ choices for a power stretch, $O(k^2)$ pairs of aligned misp. per combination. Need $\geq k$ pairs of alinged misp. Need $\geq k$ nominations.

Is this result tight?

Can we find a pattern P of length m that is not almost periodic and has $O(k^2)$ k-mismatch occurrences in a text T of length $n \leq 2m$?

Is $6k$ the right value for defining “almost periodic”?

Is this result tight?

Can we find a pattern P of length m that is not almost periodic and has $O(k^2)$ k-mismatch occurrences in a text T of length $n \leq 2m$?

Is $6k$ the right value for defining "almost periodic"?

Exercise

Construct an example where the pattern P

- is not at Hamming distance $O(k)$ from any string with period $O(m/k)$,
- has $\Omega(k)$ k-mismatch occurrences in T.

Is this result tight?

Can we find a pattern P of length m that is not almost periodic and has $O(k^2)$ k-mismatch occurrences in a text T of length $n \le 2m$?
Is $6k$ the right value for defining "almost periodic"?

Exercise

Construct an example where the pattern P

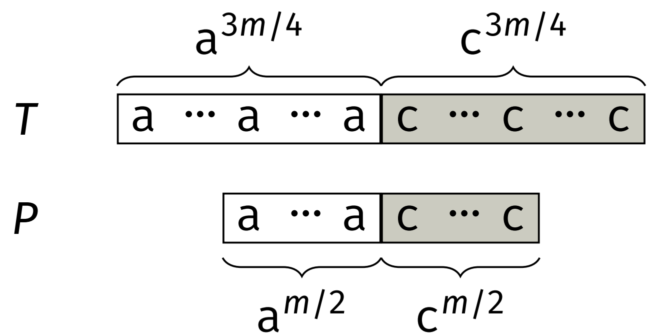
- is not at Hamming distance $O(k)$ from any string with period $O(m/k)$,
- has $\Omega(k)$ k-mismatch occurrences in T.

Exercise

Construct an example where the pattern P

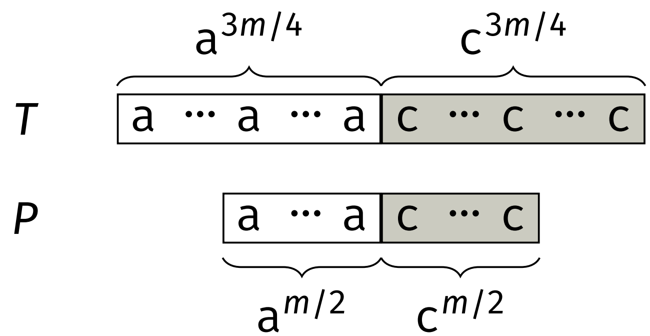
- is not periodic,
- it is at Hamming distance $O(k)$ from a string with period $O(m/k)$,
- it has $\Omega(n)$ k-mismatch occurrences in T.

Examples



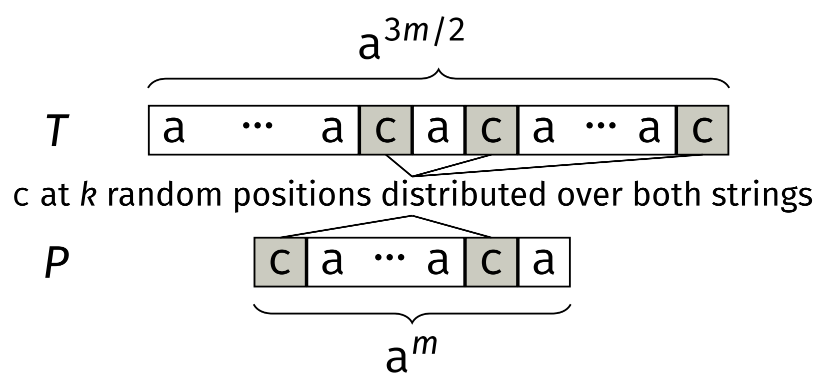
- Both P and T far from periodic, but there are $2k + 1$ k-mismatch occurrences of P in T.

Examples



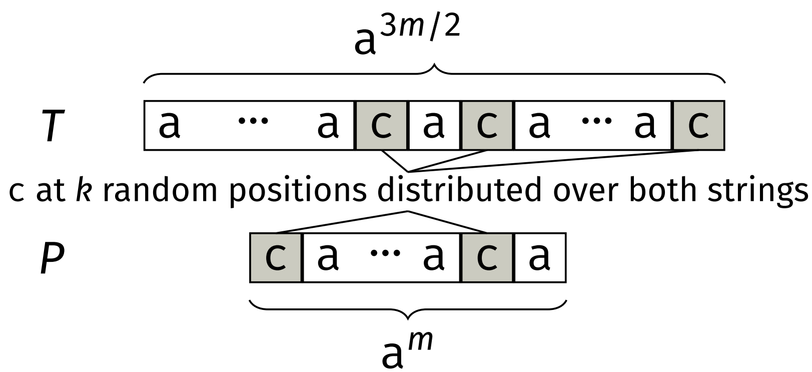
- Both P and T far from periodic, but there are $2k + 1$ k-mismatch occurrences of P in T.

Examples



- Both P and T at HD up to k from periodic, and P matches all m-length substrings of T.

Examples



- Both P and T at HD up to k from periodic, and P matches all m-length substrings of T.

Main Idea Towards an Improvement

In the approach we just saw: The pattern P is independently aligned with a substring of Q_i^∞ for each chunk P_i.

Main Idea Towards an Improvement

In the approach we just saw: The pattern P is independently aligned with a substring of Q_i^∞ for each chunk P_i.

The same position in P may be accounted for as a misperiod for multiple chunks P_i.

Main Idea Towards an Improvement

In the approach we just saw: The pattern P is independently aligned with a substring of Q_i^∞ for each chunk P_i.

The same position in P may be accounted for as a misperiod for multiple chunks P_i.

In particular, this happens if several adjacent chunks share the same period. This leads to an overcounting of the k-mismatch occurrences that is hard to control.

Main Idea Towards an Improvement

In the approach we just saw: The pattern P is independently aligned with a substring of Q_i^∞ for each chunk P_i.

The same position in P may be accounted for as a misperiod for multiple chunks P_i.

In particular, this happens if several adjacent chunks share the same period. This leads to an overcounting of the k-mismatch occurrences that is hard to control.

Idea: Analyse the (periodic structure of the) pattern as a whole.

Improved Structural Results for PM with Mismatches

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \leq {}^3/_2\, m$, and a threshold $k \leq m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $\cancel{O(k^2)}$ $O(k)$.
- The pattern P is almost periodic (at HD $\cancel{\leq 6k}$ $< 2k$ to a string Q with period $O(m/k)$).

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \leq {}^3\!/_2\, m$, and a threshold $k \leq m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD < $2k$ to a string with period $O(m/k)$).

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3\!/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD < $2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period > $m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$; each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD < $8k$ to a string with period $\le m/(128k)$).

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3\!/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD < $2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

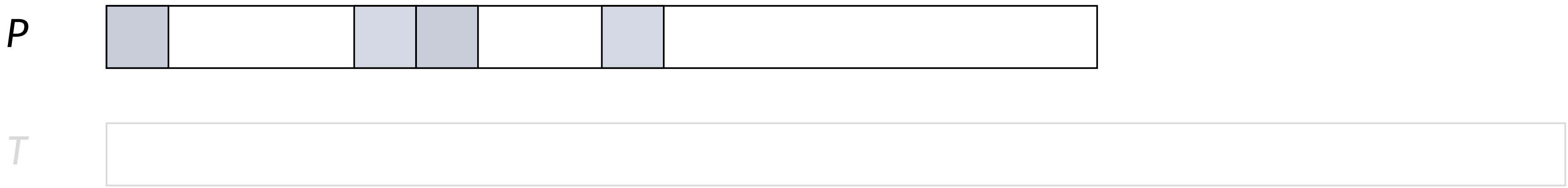
For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$; each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD < $8k$ to a string with period $\le m/(128k)$).

Observation [BKW'19, refined]

If P contains $\ge 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

Consider an example with $k = 2$.



Observation [BKW'19, refined]

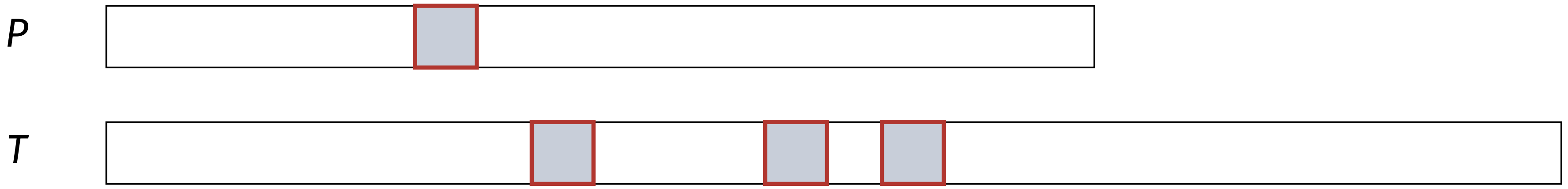
If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

Let us focus on a single break B.

Observation [BKW'19, refined]

If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

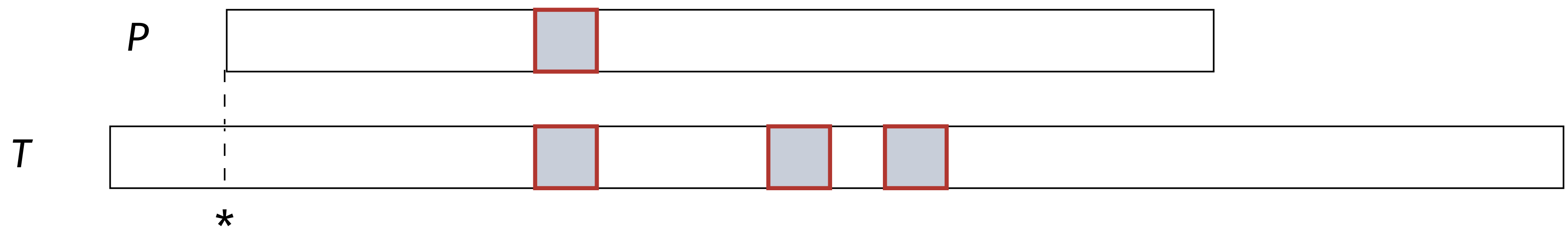
B has $O(k)$ occurrences in T since $n \leq {}^3/_2\, m$ and the period of B is $> m/(128k)$.



Observation [BKW'19, refined]

If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

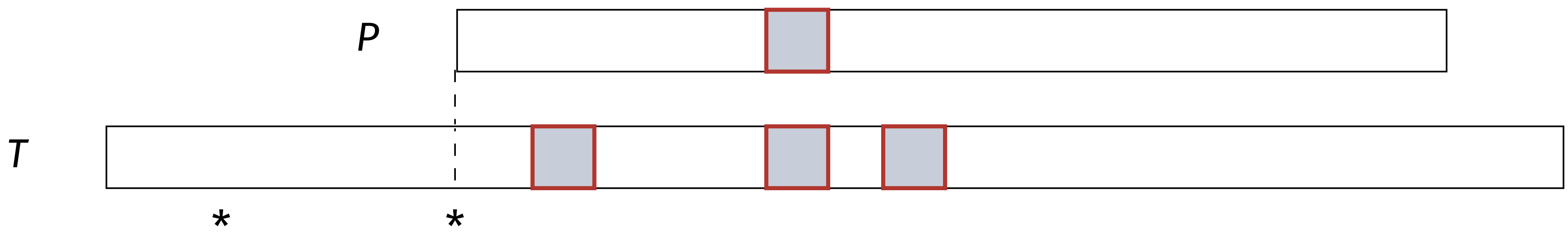
For each such occurrence, we put a mark in the position of T where P starts if we align the break with the occurrence.



Observation [BKW'19, refined]

If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

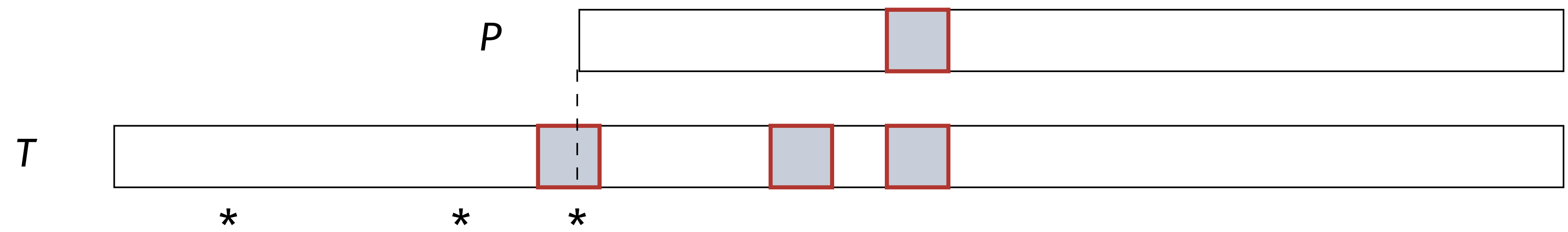
For each such occurrence, we put a mark in the position of T where P starts if we align the break with the occurrence.



Observation [BKW'19, refined]

If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

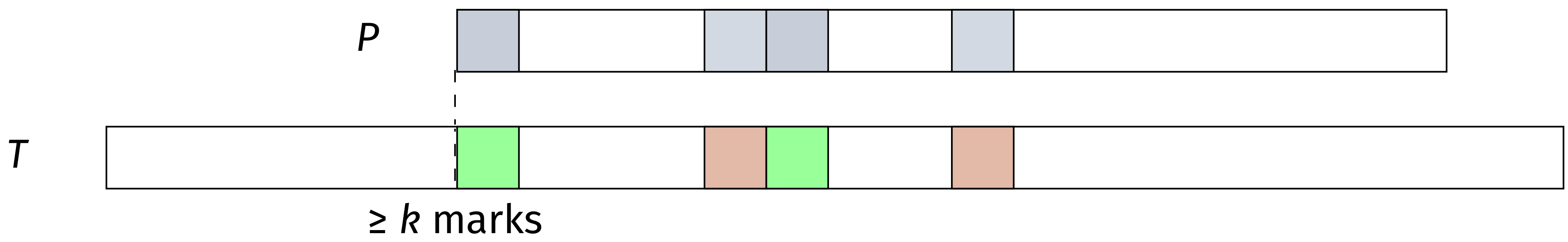
Over all breaks, we place $O(k^2)$ marks.



Observation [BKW'19, refined]

If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

Over all breaks, we place $O(k^2)$ marks. A position of T can be the starting position of a k-mismatch occurrence of P in T only if it has $\geq k$ marks.



Observation [BKW'19, refined]

If P contains $\geq 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3\!/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD $< 2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$;
  each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\le m/(128k)$).

Observation [BKW'19, refined]

If P contains $\ge 2k$ disjoint breaks, there are $O(k)$ k-mismatch occ's of P in T.

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD $< 2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

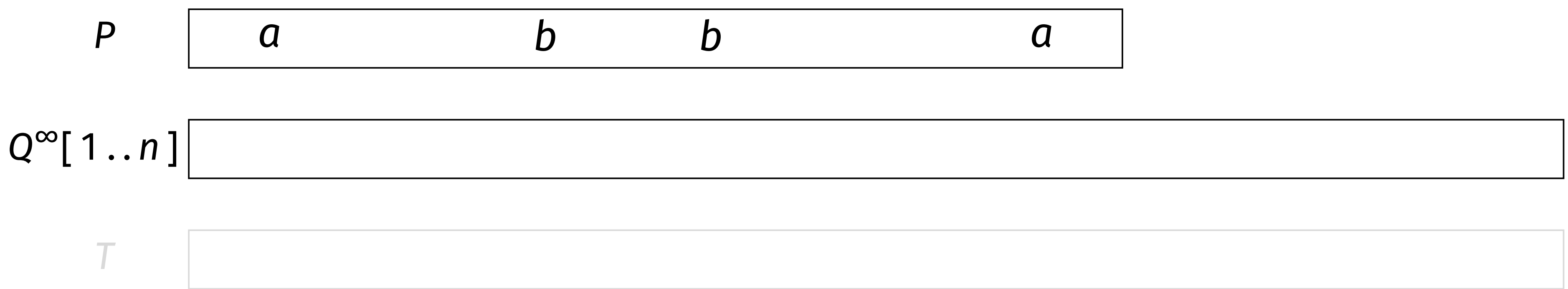
For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$;
  each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\le m/(128k)$).

Observation [BKW'19, refined]

If P has HD $\ge 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

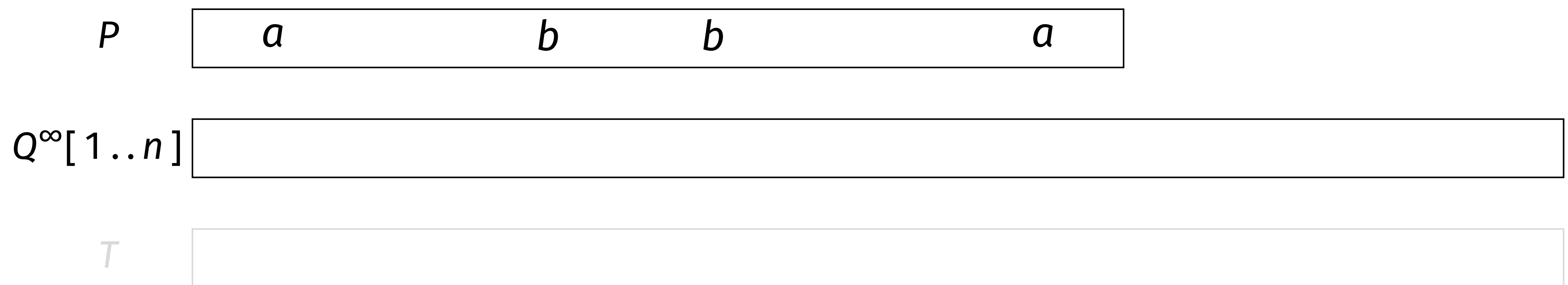
Consider again $k = 2$. We denote only the letters at positions where P differs from the length-m prefix of Q^∞, that is the misperiods.



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

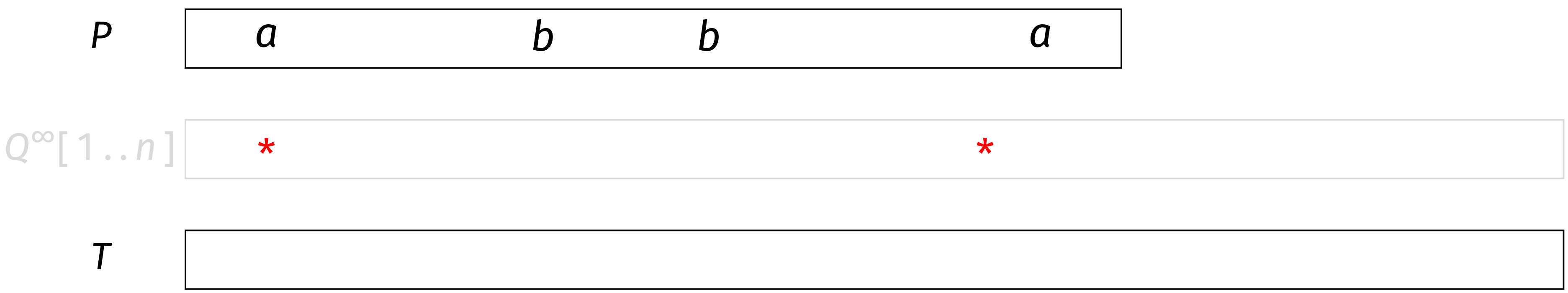
Claim: The part of T that contains all k-mismatch occurrences of P is at Hamming distance $\Theta(k)$ from the length-n prefix of Q^∞. (By the triangle inequality.)



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

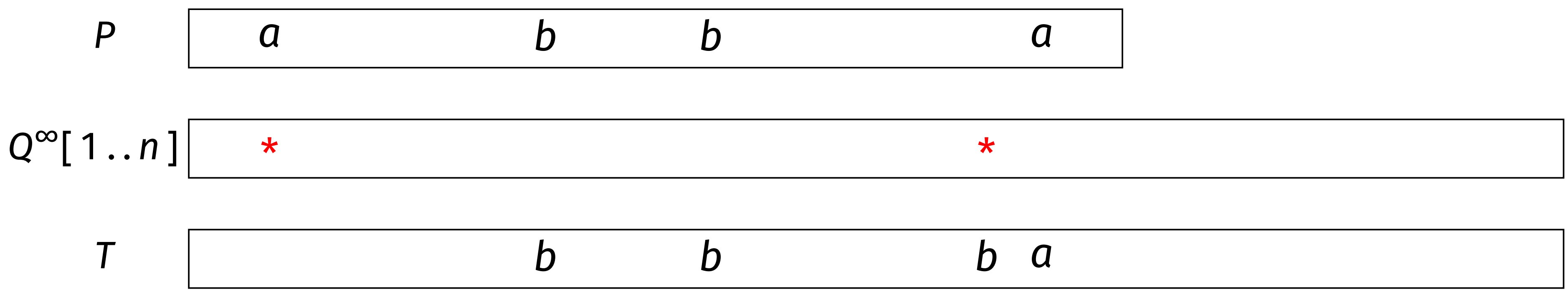
Claim: The part of T that contains all k-mismatch occurrences of P is at Hamming distance $\Theta(k)$ from the length-n prefix of Q^∞. (By the triangle inequality.)



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

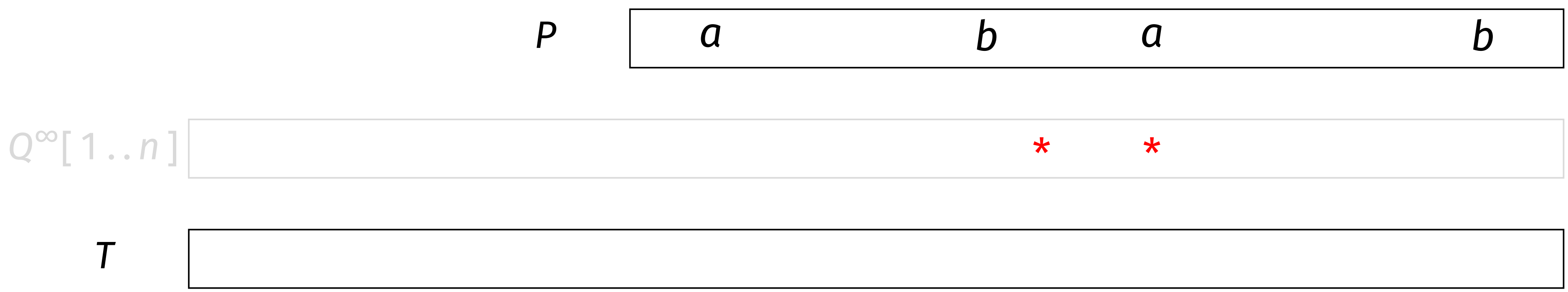
Claim: The part of T that contains all k-mismatch occurrences of P is at Hamming distance $\Theta(k)$ from the length-n prefix of Q^∞. (By the triangle inequality.)



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

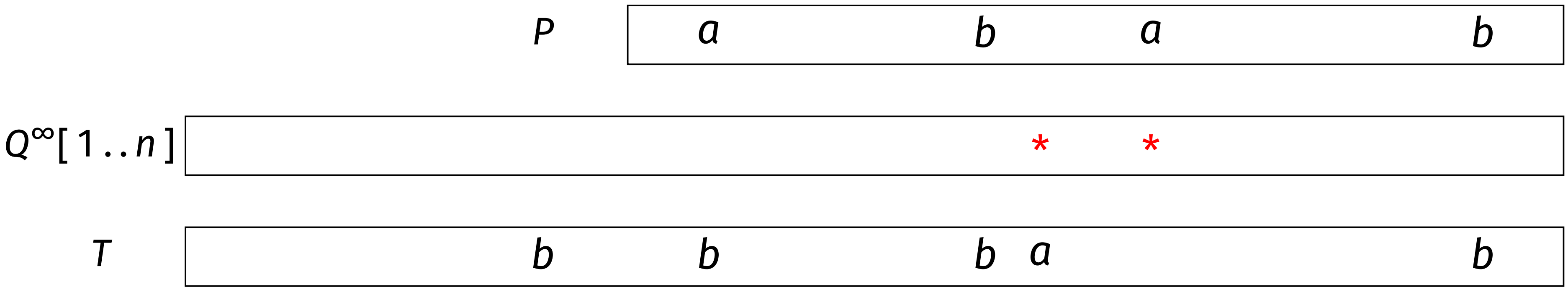
Claim: The part of T that contains all k-mismatch occurrences of P is at Hamming distance $\Theta(k)$ from the length-n prefix of Q^∞. (By the triangle inequality.)



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

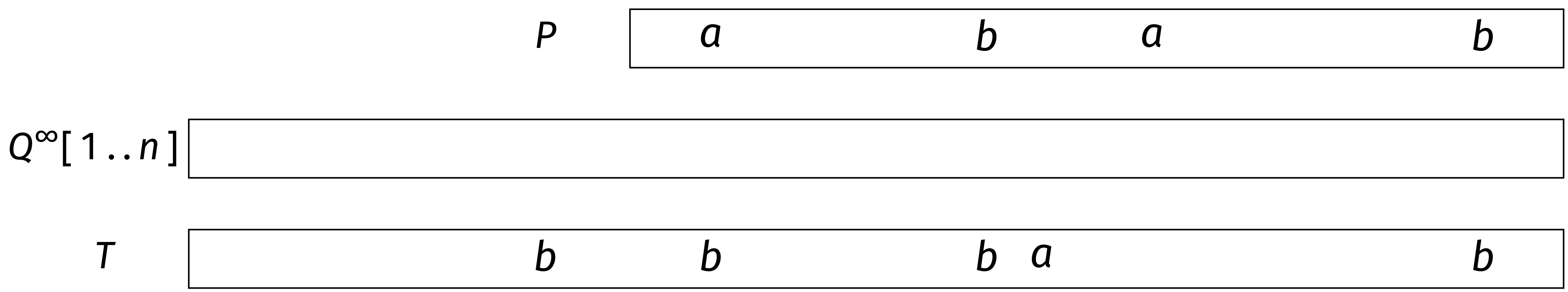
Claim: The part of T that contains all k-mismatch occurrences of P is at Hamming distance $\Theta(k)$ from the length-n prefix of Q^∞. (By the triangle inequality.)



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

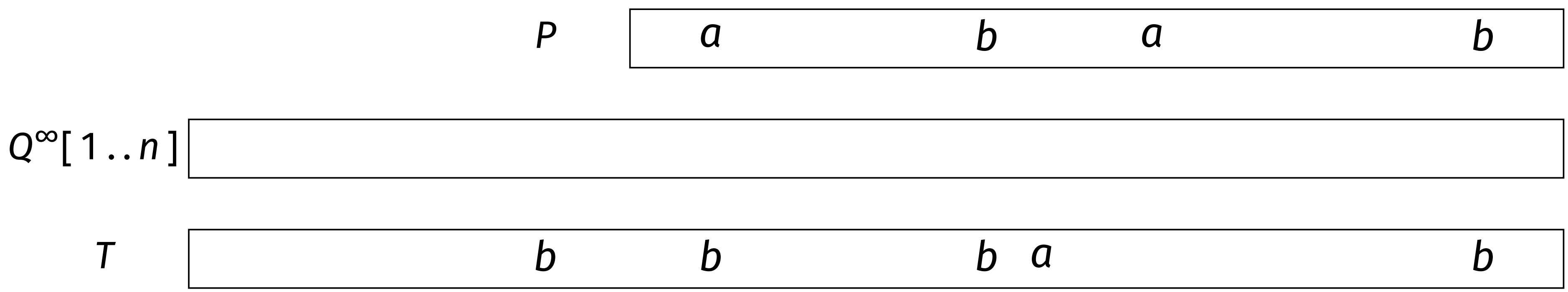
P can only have k-mismatch occurrences in positions equiv. $1 \mod |Q|$ due to periodicity.
(We have many copies of Q, and Q does not match any of its rotations.)



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

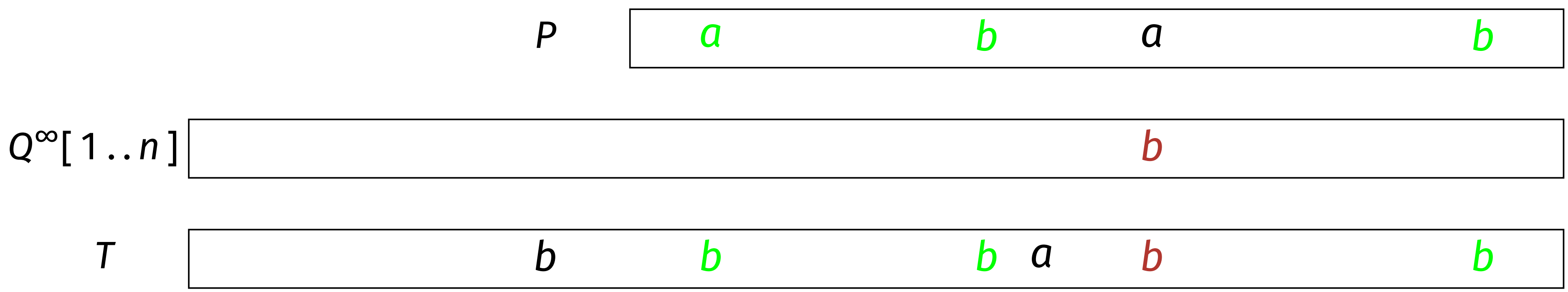
For each k-mismatch occurrence $T[i..j]$ of P, as P has $\geq 2k$ mismatches with $Q^\infty[1..m]$, at least k of P's misperiods must coincide with misperiods of $T[i..j]$.



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

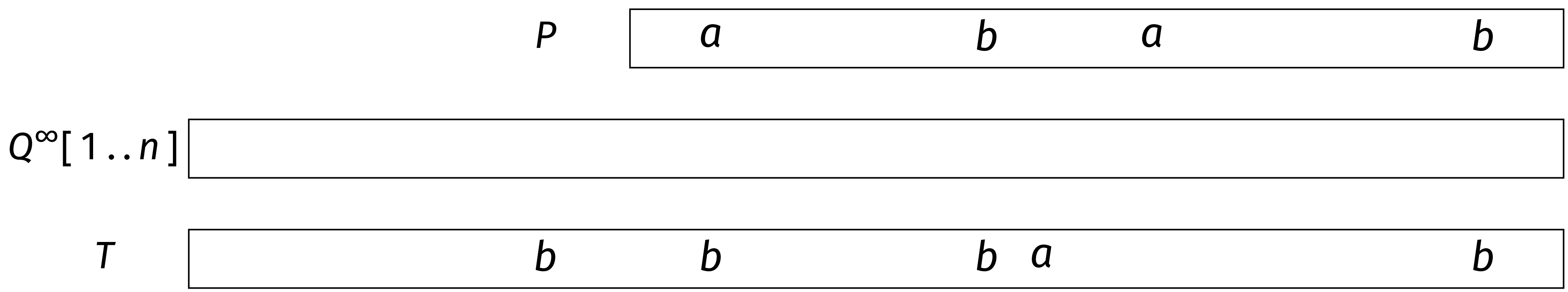
For each k-mismatch occurrence $T[i..j]$ of P, as P has $\geq 2k$ mismatches with $Q^\infty[1..m]$, at least k of P's misperiods must coincide with misperiods of $T[i..j]$.



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

As we have $\Theta(k^2)$ pairs of misperiods, we have $O(k)$ k-mismatch occurrences of P in T.



Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD $< 2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$;
  each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\le m/(128k)$).

Observation [BKW'19, refined]

If P has HD $\ge 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3\!/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD < $2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$; each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD < $8k$ to a string with period $\le m/(128k)$).

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^3\!/_2\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD < $2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\ge 3/8 \cdot m$;
  each region has length $\ge m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD < $8k$ to a string with period $\le m/(128k)$).

Observation [BKW'19, refined]

If P has HD $\ge 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \leq {}^3/_2\, m$, and a threshold $k \leq m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD $< 2k$ to a string with period $O(m/k)$).

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/128k$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$; each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).

Observation [BKW'19, refined]

If P has HD $\geq 2k$ and $< 8k$ to a string w/ period $O(m/k)$, there are $O(k)$ k-mism. occ's of P in T.

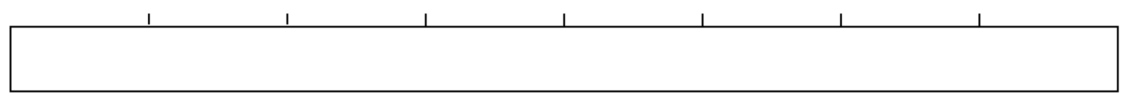
Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).

P

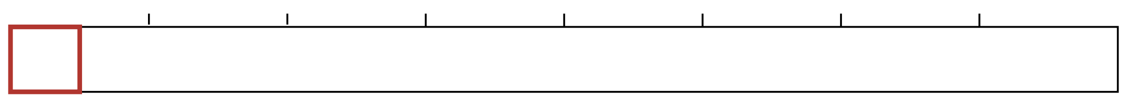


Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).

P 

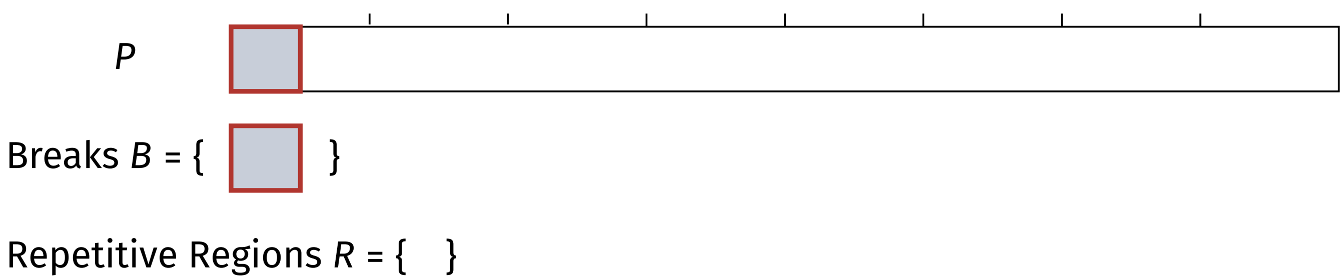
- Process P from left to right, $m/8k$ new characters at a time.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



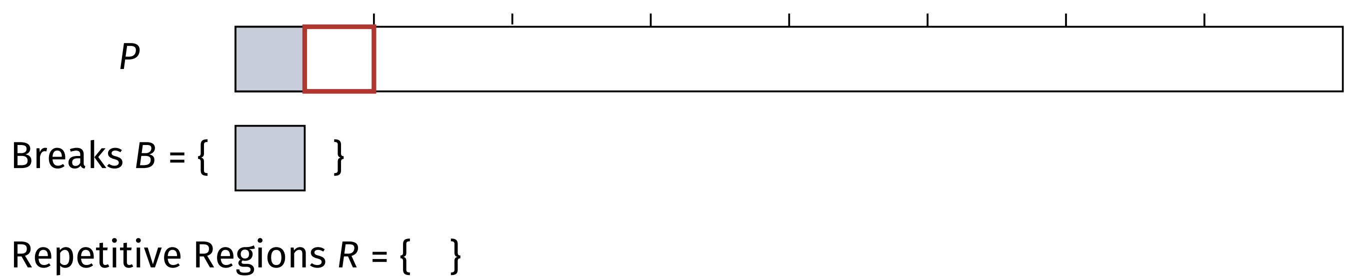
- If a fragment has a period $> m/128k$, add it to the found breaks.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



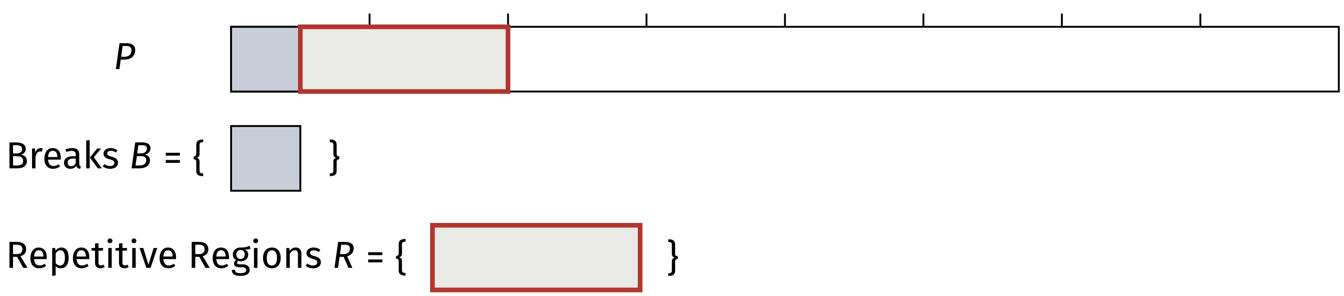
- Otherwise, find the shortest prefix (longer than $m/8k$) that is a repetitive region.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$; each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



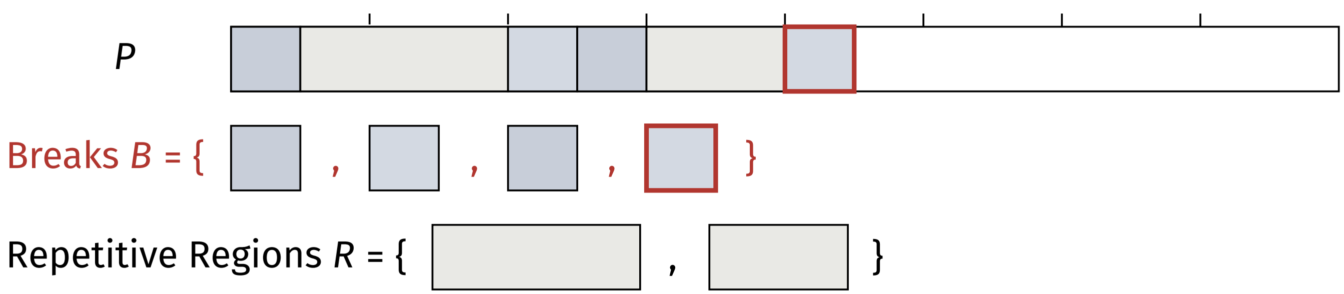
- Otherwise, find the shortest prefix (longer than $m/8k$) that is a repetitive region.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$; each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



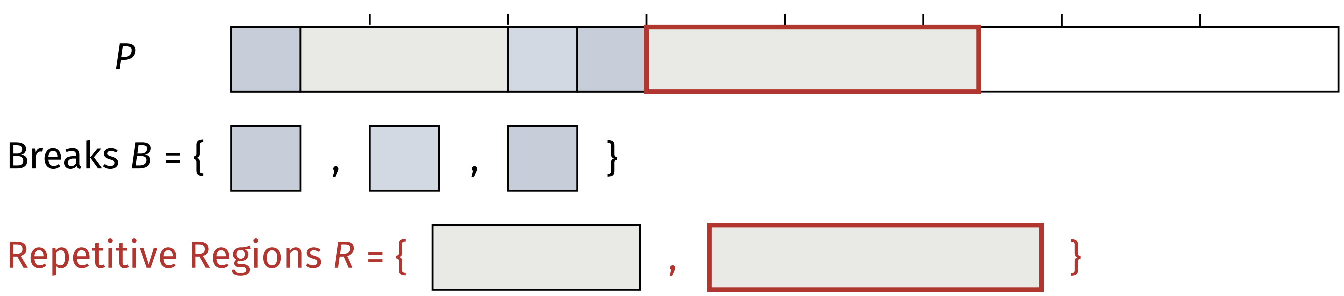
- If we found $2k$ breaks, return the breaks.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$; each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



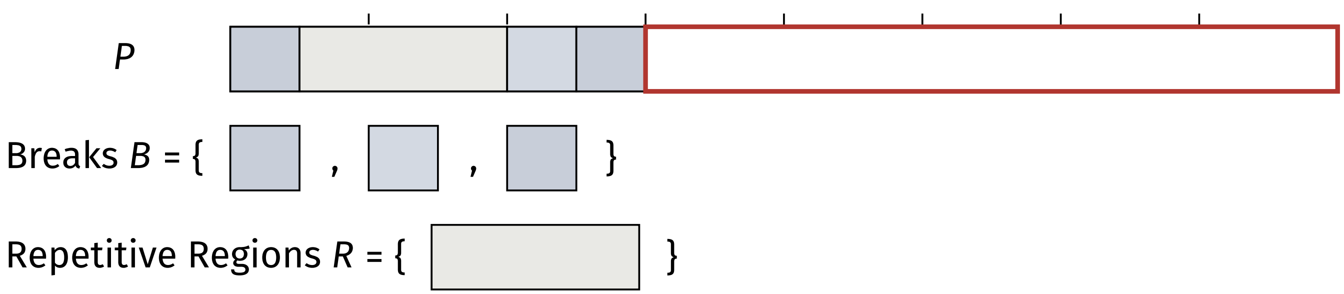
- If the total length of the repetitive regions is $> 3/8 \cdot m$, return the repetitive regions.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



- If we reach the end of P, try to find a single repetitive region starting from the end.

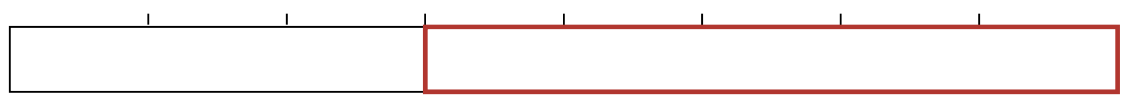
Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).

P



Breaks $B = \{ \quad \}$

Repetitive Regions $R = \{ \quad \}$

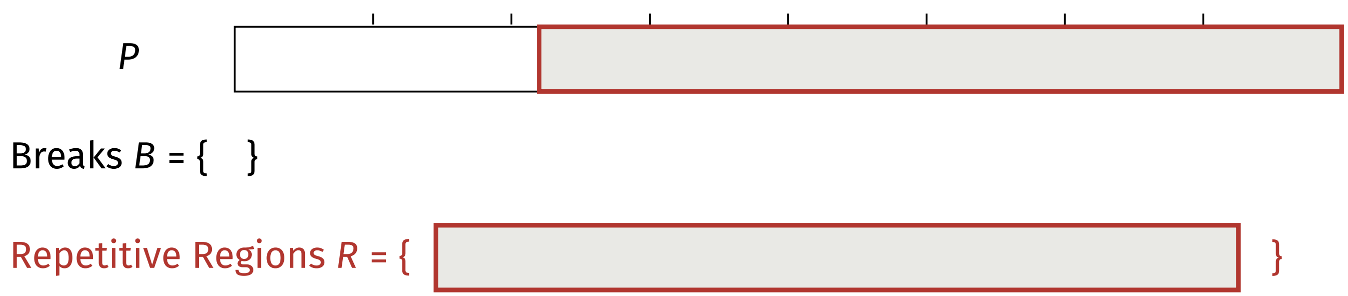
- If we reach the end of P, try to find a single repetitive region starting from the end.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$; each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).



- If we found a repetitive region, return it.

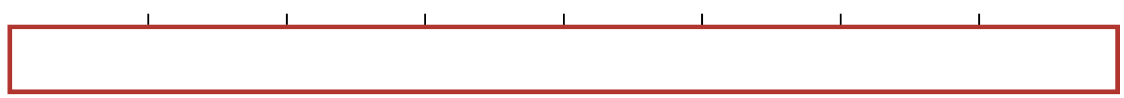
Analyzing the Pattern, Proof Idea

Key Lemma (Analyze)

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$;
  each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).

P



Breaks $B = \{\ \}$

Repetitive Regions $R = \{\ \}$

- If we again don't obtain a repetitive region, P is almost periodic.

Analyzing the Pattern, Proof Idea

Key Lemma (Analyze) ✓

For each string P of length m, at least one of the following holds:

- P contains $2k$ disjoint breaks; each break has length $m/8k$ and period $> m/(128k)$.
- P contains disjoint repetitive regions R_i with total length $\geq 3/8 \cdot m$; each region has length $\geq m/8k$ and is almost periodic with HD exactly $8k/m \cdot |R_i|$.
- P is almost periodic (at HD $< 8k$ to a string with period $\leq m/(128k)$).

What about PM with errors?

Structural Results for PM with Errors

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^{3}\!/_{2}\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD < $2k$ to a string Q with period $O(m/k)$).

Structural Theorem (ED) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \le {}^{3}\!/_{2}\, m$, and a threshold $k \le m$, at least one of the following holds:

- The number of (starting positions of) k-error occurrences of P in T is $O(k^2)$.
- The pattern P is almost periodic (at ED < $2k$ to a string Q with period $O(m/k)$).

Structural Results for PM with Errors

Structural Theorem (HD) [C.-Kociumaka-Wellnitz, FOCS 2020]

Given a pattern P of length m, a text T of length $n \leq {}^{3}\!/_{2}\, m$, and a threshold $k \leq m$, at least one of the following holds:

- The number of k-mismatch occurrences of P in T is $O(k)$.
- The pattern P is almost periodic (at HD $< 2k$ to a string Q with period $O(m/k)$).

Structural Theorem (ED) [C.-Kociumaka-Wellnitz, FOCS 2020]

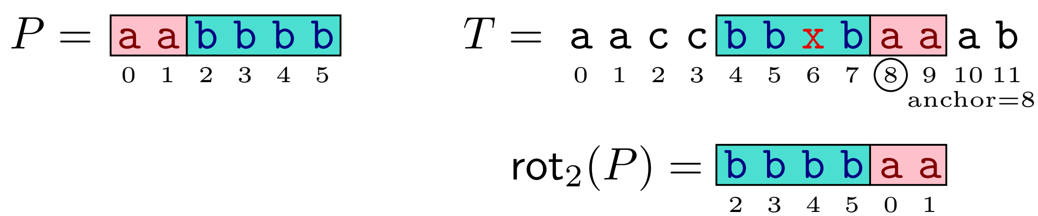
Given a pattern P of length m, a text T of length $n \leq {}^{3}\!/_{2}\, m$, and a threshold $k \leq m$, at least one of the following holds:

- The starting positions of all k-error occurrences of P in T lie in $O(k)$ intervals of length $O(k)$ each.
- The pattern P is almost periodic (at ED $< 2k$ to a string Q with period $O(m/k)$).

What next?

Consider more complicated settings.

For instance, the case where (approximately) matching any rotation of the pattern is acceptable has been already considered.



Either the pattern is almost periodic or there are $O(k)$ anchors for k-mismatch circular occurrences. (Each anchor gives $O(k)$ intervals of occurrences.)
[CKPRRSWZ, ESA'22]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]
- $O(n + k^3 \cdot n/m)$ – [Amir et al., J. Alg.'04]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]
- $O(n + k^3 \cdot n/m)$ – [Amir et al., J. Alg.'04]
- $\tilde{O}(n + k^2 \cdot n/m)$ – [Clifford et al., SODA'16]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]
- $O(n + k^3 \cdot n/m)$ – [Amir et al., J. Alg.'04]
- $\tilde{O}(n + k^2 \cdot n/m)$ – [Clifford et al., SODA'16]
- $\tilde{O}(n + kn/\sqrt{m})$ – [Gawrychowski-Uznański, ICALP'18]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]
- $O(n + k^3 \cdot n/m)$ – [Amir et al., J. Alg.'04]
- $\tilde{O}(n + k^2 \cdot n/m)$ – [Clifford et al., SODA'16]
- $\tilde{O}(n + kn/\sqrt{m})$ – [Gawrychowski-Uznański, ICALP'18]
- matching (conditional) lower bound (for combinatorial algorithms) – [G-U, ICALP'18]

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]
- $O(n + k^3 \cdot n/m)$ – [Amir et al., J. Alg.'04]
- $\tilde{O}(n + k^2 \cdot n/m)$ – [Clifford et al., SODA'16]
- $\tilde{O}(n + kn/\sqrt{m})$ – [Gawrychowski-Uznański, ICALP'18]
- matching (conditional) lower bound (for combinatorial algorithms) – [G-U, ICALP'18]
- $O(n + k^2 \cdot n/m)$ – [Chan et al. STOC'20] (improvement in log-factors, at the cost of randomisation)

Results on Approximate Pattern Matching under HD

- $O(n\sqrt{m \log m})$ – [Abrahamson, SICOMP'87], [Kosaraju '87]
- $O(nk)$ – [Landau-Vishkin, TCS'86]
- $O(n\sqrt{k \log k})$ – [Amir et al., J. Alg.'04]
- $O(n + k^3 \cdot n/m)$ – [Amir et al., J. Alg.'04]
- $\tilde{O}(n + k^2 \cdot n/m)$ – [Clifford et al., SODA'16]
- $\tilde{O}(n + kn/\sqrt{m})$ – [Gawrychowski-Uznański, ICALP'18]
- matching (conditional) lower bound (for combinatorial algorithms) – [G-U, ICALP'18]
- $O(n + k^2 \cdot n/m)$ – [Chan et al. STOC'20] (improvement in log-factors, at the cost of randomisation)
- $\tilde{O}(n + k^2 \cdot n/m)$ – [CKW'20] (improvement in log-factors)

How do we turn the structural insights into algorithms?

Obtaining Faster Algorithms

- Create algorithms that rely on a small set of essential operations:
  - `LCP`(S, T): Compute the length of the longest common prefix of S and T.
  - `LCP`^{R}(S, T): Compute the length of the longest common suffix of S and T.
  - `IPM`(P, T): Compute all exact matches of P in T.
  - `Length`(S): Compute the length $|S|$ of S.
  - `Access`(S, i): Retrieve the character $S[i]$.
  - `Extract`(S, ℓ, r): Extract the fragment (or substring) $S[\ell\,..\,r)$ from S.

The `PILLAR` Model

- Create algorithms that rely on a small set of essential operations:
  - `LCP`(S, T): Compute the length of the longest common prefix of S and T.
  - `LCP`^{R}(S, T): Compute the length of the longest common suffix of S and T.
  - `IPM`(P, T): Compute all exact matches of P in T.
  - `Length`(S): Compute the length $|S|$ of S.
  - `Access`(S, i): Retrieve the character $S[i]$.
  - `ExtRact`(S, ℓ, r): Extract the fragment (or substring) $S[\ell \,..\, r)$ from S.

The PILLAR Model

- Create algorithms that rely on a small set of essential operations:
  - LC**P**(S, T): Compute the length of the longest common prefix of S and T.
  - **L**CP$^R(S, T)$: Compute the length of the longest common suffix of S and T.
  - **I**PM(P, T): Compute all exact matches of P in T.
  - **L**ength(S): Compute the length $|S|$ of S.
  - **A**ccess(S, i): Retrieve the character $S[i]$.
  - Ext**R**act(S, ℓ, r): Extract the fragment (or substring) $S[\ell\,..\,r)$ from S.

Theorem
Fast PILLAR Algorithm

+

Theorem
Fast PILLAR Implementation

=

Theorem
Fast Algorithm

An algorithm for the almost periodic case

Consider a pattern P that is at Hamming distance $< 2k$ from a prefix of Q^∞, where Q is primitive and $|Q| \le m/(128k)$, and a string P that is at Hamming distance $< 6k$ from a prefix of Q^∞.

Suppose that we are given the $O(k)$ misperiods for each of P and T.

How fast can we compute a representation of k-mismatch occurrences of P in T?

Hint 1: The exact occurrences of $(\mathrm{ab})^{70}$ in $(\mathrm{ab})^{100}$ can be represented as a single arithmetic progression $\{1 + 2i : i \in [0, 30]\}$.

Hint 2: k-mismatch occurrences of P can only start at positions of T that are $\equiv 1 \bmod |Q|$.

What Changes for Edit Distance?

Brief discussion on the board.

The PILLAR Model: Fast PILLAR Algorithms

The PILLAR operations: LCP, LCP^R, IPM, Length, Access, Extract

Theorem (PILLAR Alg. for PM w/ Mism.)

Given a pattern P of length m, a text T of length n, and a positive threshold $k \le m$, we can compute (a representation of) all k-mismatch occurrences of P in T using $O(n/m \cdot k^2 \log\log k)$ time plus $O(n/m \cdot k^2)$ PILLAR operations.

The PILLAR Model: Fast PILLAR Algorithms

The PILLAR operations: LCP, LCP^R, IPM, Length, Access, Extract

Theorem (PILLAR Alg. for PM w/ Mism.)

Given a pattern P of length m, a text T of length n, and a positive threshold $k \le m$, we can compute (a representation of) all k-mismatch occurrences of P in T using $O(n/m \cdot k^2 \log\log k)$ time plus $O(n/m \cdot k^2)$ PILLAR operations.

Theorem (PILLAR Alg. for PM w/ Errors)

Given a pattern P of length m, a text T of length n, and a positive threshold $k \le m$, we can compute (a representation of) all k-error occurrences of P in T using $\tilde{O}(n/m \cdot k^{3.5})$ PILLAR operations.

The PILLAR Model: The Standard Setting (HD)

The PILLAR operations: LCP, LCP^R, IPM, Length, Access, Extract
Uncompressed strings: pattern P of length m, text T of length n.

The $\texttt{PILLAR}$ Model: The Standard Setting (HD)

The $\texttt{PILLAR}$ operations: $\texttt{LCP}$, $\texttt{LCP}^R$, $\texttt{IPM}$, $\texttt{Length}$, $\texttt{Access}$, $\texttt{Extract}$

Uncompressed strings: pattern P of length m, text T of length n.

We can perform each operation in $O(1)$ time. (After $O(n + m)$-time preprocessing.)

The PILLAR Model: The Standard Setting (HD)

The PILLAR operations: LCP, LCP^R, IPM, Length, Access, Extract

Uncompressed strings: pattern P of length m, text T of length n.

We can perform each operation in $O(1)$ time. (After $O(n+m)$-time preprocessing.)

Theorem (Algorithm for PM w/ Mism.)

For any positive threshold $k \le m$,
we can compute all k-mismatch occurrences of P in T in time $O(n + n/m \cdot k^2 \log\log k)$.

What if the text and the pattern are *huge*?

o	f	s	w	e	e	t	b	e	e	n	c	e	n	t	e	r	e	d	i	n	t	h	e	a	n	p	a	n	f	o	r	e	x	a	m

n	p	a	s	t	e	a	n	p	a	n	c	a	n	a	l	s	o	b	e	p	r	e	p	a	r	e	d	w	i	t	h	o	t	h	e

What if the text and the pattern are *huge*?

anpanisoneofthepopularjapanesesweetbunwithsweetbeaninthecentertodaytherearemanytypesofsweetbeencenteredintheanpanforexamplegomaanshiroanuguisuankurianandetcbuttheoriginalofitisthenormalankomadewithredbean

anpanisajapanesesweetrollmostcommonlyfilledwithredbeanpasteanpancanalsobepreparedwithotherfillingsincludingwhitebeansgreenbeanssesameandchestnut

What if the text and the pattern are given in a compressed representation?

anpanisoneofthepopularjapanesesweetbunwithsweetbeaninthecentertodaytherearemanytypesofsweetbeencenteredintheanpanforexamplegomaanshiroanuguisuankurianandetcbuttheoriginalofitisthenormalankomadewithredbean

anpanisajapanesesweetrollmostcommonlyfilledwithredbeanpasteanpancanalsobepreparedwithotherfillingsincludingwhitebeansgreenbeanssesameandchestnut

Grammar Compression

Grammar Compression

For a string T, a grammar compression of T is a context-free grammar G_T that generates $\{T\}$. The grammar G_T is wlog. a straight-line program or SLP.

Grammar Compression

Straight-Line Program (SLP)

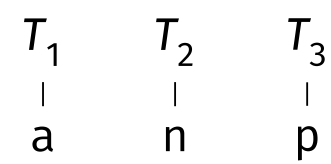
An SLP G_T is a set of non-terminals $\{T_1, \dots, T_n\}$ and productions of the form $T_i \rightarrow a, a \in \Sigma$ or $T_i \rightarrow T_\ell T_r$, where $\ell, r < i$. The starting symbol is T_n.

Grammar Compression

Straight-Line Program (SLP)

An SLP G_T is a set of non-terminals $\{T_1, \ldots, T_n\}$ and productions of the form $T_i \to a, a \in \Sigma$ or $T_i \to T_\ell T_r$, where $\ell, r < i$. The starting symbol is T_n.

$T_1 \to \text{a}; \quad T_2 \to \text{n}; \quad T_3 \to \text{p}$



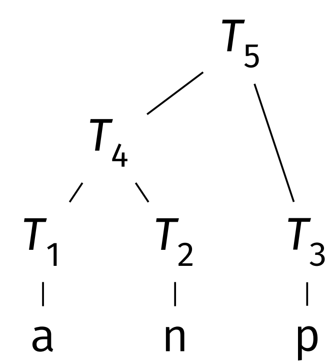
Grammar Compression

Straight-Line Program (SLP)

An SLP G_T is a set of non-terminals $\{T_1, \ldots, T_n\}$ and productions of the form $T_i \rightarrow a, a \in \Sigma$ or $T_i \rightarrow T_\ell T_r$, where $\ell, r < i$. The starting symbol is T_n.

$$T_1 \rightarrow \mathrm{a}; \quad T_2 \rightarrow \mathrm{n}; \quad T_3 \rightarrow \mathrm{p}$$

$$T_4 \rightarrow T_1 T_2; \qquad T_5 \rightarrow T_4 T_3$$



Grammar Compression

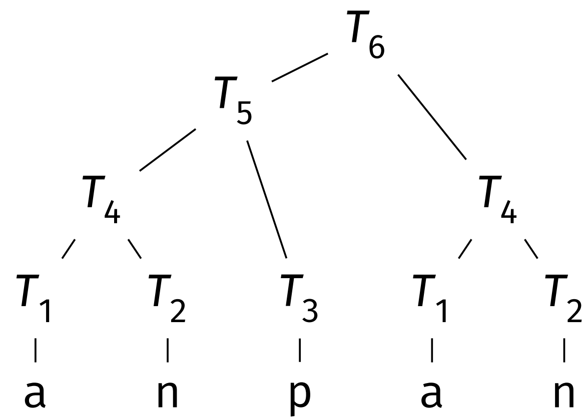
Straight-Line Program (SLP)

An SLP G_T is a set of non-terminals $\{T_1, \ldots, T_n\}$ and productions of the form $T_i \to a, a \in \Sigma$ or $T_i \to T_\ell T_r$, where $\ell, r < i$. The starting symbol is T_n.

$$T_1 \to \mathrm{a}; \quad T_2 \to \mathrm{n}; \quad T_3 \to \mathrm{p}$$

$$T_4 \to T_1 T_2; \qquad T_5 \to T_4 T_3$$

$$T_6 \to T_5 T_4$$



Grammar Compression

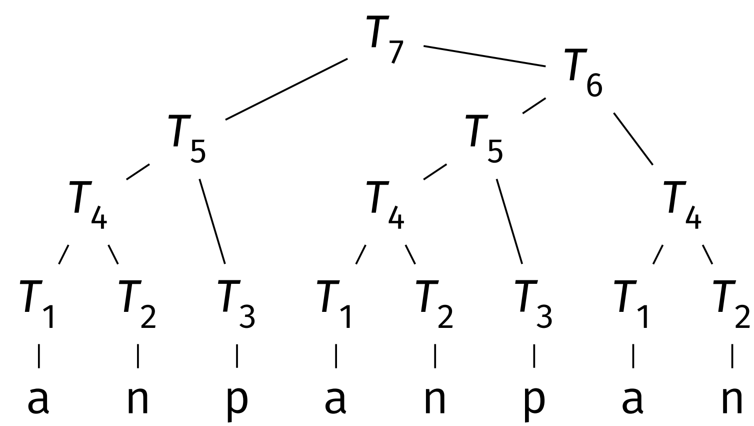
Straight-Line Program (SLP)

An SLP G_T is a set of non-terminals $\{T_1, \ldots, T_n\}$ and productions of the form $T_i \to a, a \in \Sigma$ or $T_i \to T_\ell T_r$, where $\ell, r < i$. The starting symbol is T_n.

$$T_1 \to \mathrm{a}; \quad T_2 \to \mathrm{n}; \quad T_3 \to \mathrm{p}$$

$$T_4 \to T_1 T_2; \qquad T_5 \to T_4 T_3$$

$$T_6 \to T_5 T_4; \qquad T_7 \to T_5 T_6$$



Faster Algorithms

Problem	uncompressed text and pattern	SLP text and pattern $n = \Omega(\log N)$, $m = \Omega(\log M)$
Pattern Matching	$O(N + M)$ [KMP'77]	$\tilde{O}(n + m)$ [Jeż'15]
PM with k Mismatches	$\tilde{O}(N + k^2 \cdot N/M)$, $\tilde{O}(N + kN/\sqrt{M})$ [CFPSS'16] [GU'18]	$\tilde{O}(nk^4 + Mk)$ [BKW'19]
PM with k Errors	$O(N + k^4 \cdot N/M)$ [CH'02]	$O(nm\,\mathrm{poly}(k))$ [BLRSSW'15]

N: length of uncompressed text

n: size of compressed text

k: number of mismatches/errors

M: length of uncompressed pattern

m: size of compressed pattern

Faster Algorithms

Problem	uncompressed text and pattern	SLP text and pattern $n = \Omega(\log N)$, $m = \Omega(\log M)$
Pattern Matching	$O(N + M)$ [KMP'77]	$\tilde{O}(n + m)$ [Jeż'15]
PM with k Mismatches	$\tilde{O}(N + k^2 \cdot N/M)$, $\tilde{O}(N + kN/\sqrt{M})$ [CFPSS'16] [GU'18]	$\tilde{O}(nk^4 + Mk)$ [BKW'19]
PM with k Errors	$O(N + k^4 \cdot N/M)$ [CH'02]	$O(nm \operatorname{poly}(k))$ [BLRSSW'15]

N: length of uncompressed text
n: size of compressed text
k: number of mismatches/errors

M: length of uncompressed pattern
m: size of compressed pattern

Faster Algorithms

Problem	uncompressed text and pattern	SLP text and pattern $n = \Omega(\log N)$, $m = \Omega(\log M)$
Pattern Matching	$O(N + M)$ [KMP'77]	$\tilde{O}(n + m)$ [Jeż'15]
PM with k Mismatches	$\tilde{O}(N + k^2 \cdot N/M)$, $\tilde{O}(N + kN/\sqrt{M})$ [CFPSS'16] [GU'18]	$\tilde{O}(nk^4 + Mk)$ [BKW'19]
PM with k Errors	$O(N + k^4 \cdot N/M)$ [CH'02]	$O(nm \operatorname{poly}(k))$ [BLRSSW'15]

N: length of uncompressed text
n: size of compressed text
k: number of mismatches/errors

M: length of uncompressed pattern
m: size of compressed pattern

Faster Algorithms

Problem	uncompressed text and pattern	SLP text and pattern $n = \Omega(\log N)$, $m = \Omega(\log M)$
Pattern Matching	$O(N + M)$ [KMP'77]	$\tilde{O}(n + m)$ [Jeż'15]
PM with k Mismatches	$\tilde{O}(N + k^2 \cdot N/M)$, $\tilde{O}(N + kN/\sqrt{M})$ [CFPSS'16] [GU'18]	$\cancel{\tilde{O}(nk^4 + Mk)}$ $\tilde{O}(nk^2 + m)$
PM with k Errors	$O(N + k^4 \cdot N/M)$ [CH'02]	$\cancel{O(nm\,\mathrm{poly}(k))}$ $\tilde{O}(nk^{3.5} + m)$

N: length of uncompressed text

n: size of compressed text

k: number of mismatches/errors

M: length of uncompressed pattern

m: size of compressed pattern

Faster Algorithms

Problem	uncompressed text and pattern	SLP text and pattern $n = \Omega(\log N)$, $m = \Omega(\log M)$
Pattern Matching	$O(N + M)$ [KMP'77]	$\tilde{O}(n + m)$ [Jeż'15]
PM with k Mismatches	$\tilde{O}(N + k^2 \cdot N/M)$, $\tilde{O}(N + kN/\sqrt{M})$ $\tilde{O}(N + k^2 \cdot N/M)$	$\cancel{\tilde{O}(nk^4 + Mk)}$ $\tilde{O}(nk^2 + m)$
PM with k Errors	$O(N + k^4 \cdot N/M)$ $\tilde{O}(N + k^{3.5} \cdot N/M)$	$\cancel{O(nm\,\mathrm{poly}(k))}$ $\tilde{O}(nk^{3.5} + m)$

N: length of uncompressed text

M: length of uncompressed pattern

n: size of compressed text

m: size of compressed pattern

k: number of mismatches/errors

Faster Algorithms

Problem	uncompressed text and pattern	SLP text and pattern $n = \Omega(\log N)$, $m = \Omega(\log M)$
Pattern Matching	$O(N + M)$ [KMP'77]	$\tilde{O}(n + m)$ [Jeż'15]
PM with k Mismatches	$\tilde{O}(N + k^2 \cdot N/M)$, $\tilde{O}(N + kN/\sqrt{M})$ $\tilde{O}(N + k^2 \cdot N/M)$	$\cancel{\tilde{O}(nk^4 + Mk)}$ $\tilde{O}(nk^2 + m)$
PM with k Errors	$O(N + k^4 \cdot N/M)$ $\tilde{O}(N + k^{3.5} \cdot N/M)$	$\cancel{O(nm\,\mathrm{poly}(k))}$ $\tilde{O}(nk^{3.5} + m)$

Improvements obtained via improved/new structural insights in solution structure.

Known Results: The Fully-Compressed Setting (HD)

The `PILLAR` operations: `LCP`, $\texttt{LCP}^R$, `IPM`, `Length`, `Access`, `Extract`

SLPs: G_P of size m generating pattern P, G_T of size n generating text T.

Known Results: The Fully-Compressed Setting (HD)

The PILLAR operations: LCP, LCP^R, IPM, Length, Access, Extract

SLPs: G_P of size m generating pattern P, G_T of size n generating text T.

Using Recompression [Jeż'15], we can implement each operation in $O(\log^3(|P| + |T|))$ time.

(After $O((n + m) \log(|P| + |T|))$ preprocessing.)

Known Results: The Fully-Compressed Setting (HD)

The PILLAR operations: LCP, LCP^R, IPM, Length, Access, Extract

SLPs: G_P of size m generating pattern P, G_T of size n generating text T.

Using Recompression [Jeż'15], we can implement each operation in $O(\log^3(|P| + |T|))$ time.

(After $O((n+m)\log(|P| + |T|))$ preprocessing.)

Theorem (Algorithm for PM w/ Mism.)

For any positive threshold $k \le |P|$, we can compute the number of all k-mismatch occ's of P in T in time $O(m \log(|P| + |T|) + nk^2 \log^3(|P| + |T|))$.

(Reporting of all occurrences takes time linear in the number of occurrencess.)

Known Results: The Fully-Compressed Setting (HD)

The `PILLAR` operations: `LCP`, $\texttt{LCP}^R$, `IPM`, `Length`, `Access`, `Extract`

SLPs: G_P of size m generating pattern P, G_T of size n generating text T.

Using Recompression [Jeż'15], we can implement each operation in $O(\log^3(|P| + |T|))$ time.

(After $O((n+m)\log(|P| + |T|))$ preprocessing.)

Theorem (Algorithm for PM w/ Mism.)

For any positive threshold $k \le |P|$, we can compute the number of all k-mismatch occ's of P in T in time $O(m\log(|P| + |T|) + nk^2\log^3(|P| + |T|))$.

(Reporting of all occurrences takes time linear in the number of occurrencess.)

Let us see how on the board!

Known Results: The Dynamic Setting (HD)

The `PILLAR` operations: `LCP`, `LCP`^{R}, `IPM`, `Length`, `Access`, `Extract`

Dynamic maintenance of a collection of (non-empty persistent) strings X of total length N; supporting `makestring`, `concat`, `split`.

Known Results: The Dynamic Setting (HD)

The `PILLAR` operations: `LCP`, $\texttt{LCP}^R$, `IPM`, `Length`, `Access`, `Extract`

Dynamic maintenance of a collection of (non-empty persistent) strings X of total length N; supporting `makestring`, `concat`, `split`.

Using Optimal Dynamic Strings [Gawrychowski et al, SODA 2018], we can implement each `PILLAR` operation in $O(\log^2 N)$ time (w.h.p).

Known Results: The Dynamic Setting (HD)

The `PILLAR` operations: `LCP`, $\texttt{LCP}^R$, `IPM`, `Length`, `Access`, `Extract`

Dynamic maintenance of a collection of (non-empty persistent) strings X of total length N; supporting `makestring`, `concat`, `split`.

Using Optimal Dynamic Strings [Gawrychowski et al, SODA 2018], we can implement each `PILLAR` operation in $O(\log^2 N)$ time (w.h.p).

Theorem (Algorithm for PM w/ Mism.)

For any two strings $P, T \in X$ and any threshold k, we support the additional operation "Find all k-mismatch occ's of P in T" in $O(|T|/|P| \cdot k^2 \log^2 N)$ time (w.h.p).

Known Results: The Quantum Setting

Exercise

`PILLAR` operations can be performed in roughly $O(\sqrt{n})$ time by a quantum computer (without any preprocessing). How fast can we solve approximate pattern matching under each of the two studied distances in the quantum setting using the results we have already seen?

Known Results: The Quantum Setting

Exercise

PILLAR operations can be performed in roughly $O(\sqrt{n})$ time by a quantum computer (without any preprocessing). How fast can we solve approximate pattern matching under each of the two studied distances in the quantum setting using the results we have already seen?

For HD, an $\tilde{O}(k\sqrt{n})$-time algorithm is known [Jin-Nogler, SODA 2023].

Longest common extension (LCE) queries (standard setting)

Longest common extension (LCE) queries (standard setting)

Longest common prefix

Input: A string S of length n.

Query: Given positions i and j, compute the length of the longest common prefix of $S[i..n]$ and $S[j..n]$.

Longest common extension (LCE) queries (standard setting)

Longest common prefix

Input: A string S of length n.

Query: Given positions i and j, compute the length of the longest common prefix of $S[i..n]$ and $S[j..n]$.

Suffix tree [McCreight, JACM'76] and Lowest common ancestors [Harel and Tarjan, SICOMP'84] $\Rightarrow$ space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(1)$.

Longest common extension (LCE) queries (standard setting)

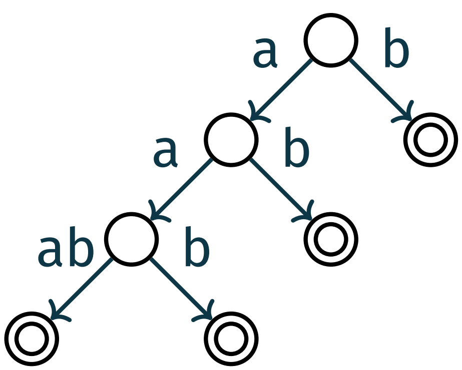
Longest common prefix

Input: A string S of length n.

Query: Given positions i and j, compute the length of the longest common prefix of $S[i..n]$ and $S[j..n]$.

Suffix tree [McCreight, JACM'76] and Lowest common ancestors [Harel and Tarjan, SICOMP'84] $\Rightarrow$ space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(1)$.

The suffix tree of $S = \mathtt{aaab}$.



Longest common extension (LCE) queries (standard setting)

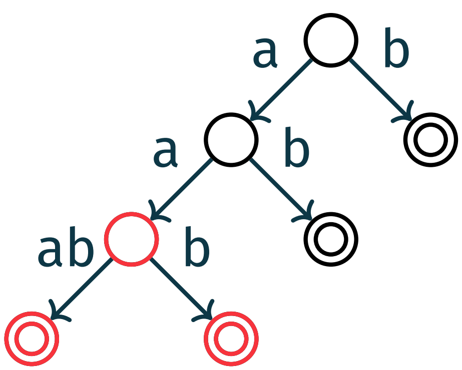
Longest common prefix

Input: A string S of length n.

Query: Given positions i and j, compute the length of the longest common prefix of $S[i\,..\,n]$ and $S[j\,..\,n]$.

Suffix tree [McCreight, JACM'76] and Lowest common ancestors [Harel and Tarjan, SICOMP'84] $\Rightarrow$ space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(1)$.

The suffix tree of $S = \mathtt{aaab}$.



The longest common prefix of $S[1\,..\,4]$ and $S[2\,..\,4]$ is the string spelled on the path from the root to the lowest common ancestor of the two nodes "representing" these substrings.

Internal pattern matching queries (standard setting)

Internal pattern matching queries (standard setting)

Internal Pattern Matching

Input: A string S of length n.

Query: Compute the occurrences of a substring U of S in another substring V of S.

Internal pattern matching queries (standard setting)

Internal Pattern Matching

Input: A string S of length n.

Query: Compute the occurrences of a substring U of S in another substring V of S.

Keller et al. [TCS 2014]: space $\tilde{\mathcal{O}}(n)$, $t_{query} = \mathcal{O}(\log\log n + |\text{output}|)$.

Internal pattern matching queries (standard setting)

Internal Pattern Matching

Input: A string S of length n.

Query: Compute the occurrences of a substring U of S in another substring V of S.

Keller et al. [TCS 2014]: space $\tilde{\mathcal{O}}(n)$, $t_{query} = \mathcal{O}(\log\log n + |\text{output}|)$.

Kociumaka et al. [SODA 2015]: space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(|V|/|U|)$. $\leftarrow$ heavily exploits the periodic structure of S

Internal pattern matching queries (standard setting)

Internal Pattern Matching

Input: A string S of length n.

Query: Compute the occurrences of a substring U of S in another substring V of S.

Keller et al. [TCS 2014]: space $\tilde{\mathcal{O}}(n)$, $t_{query} = \mathcal{O}(\log \log n + |\text{output}|)$.

Kociumaka et al. [SODA 2015]: space $\mathcal{O}(n)$, $t_{query} = \mathcal{O}(|V|/|U|)$. $\leftarrow$ heavily exploits the periodic structure of S

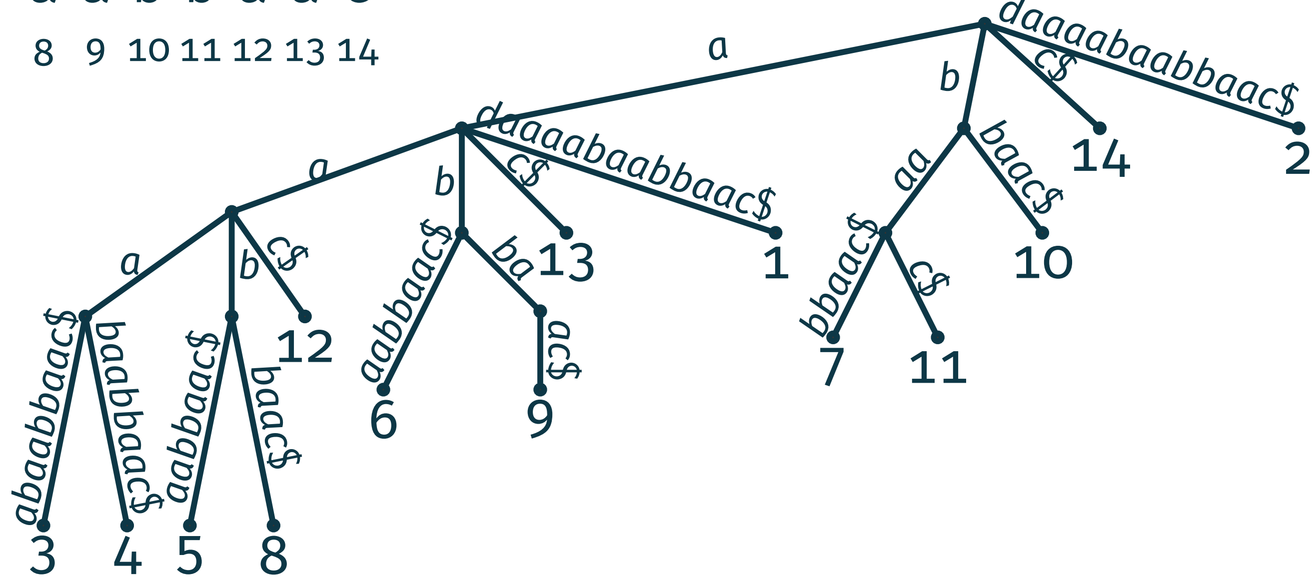
We will see a very cool reduction due to Mäkinen and Navarro [LATIN 2006].

IPM queries $\rightarrow$ 2D Range Reporting

T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

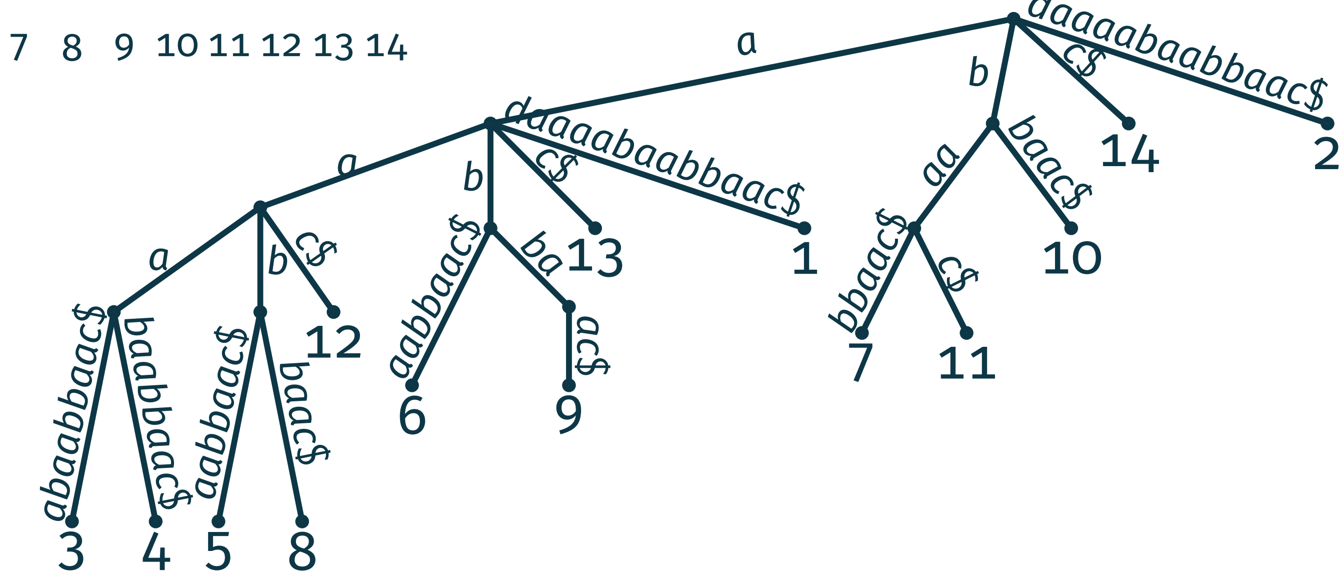
IPM queries → 2D Range Reporting

T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c
	1	2	3	4	5	6	7	8	9	10	11	12	13	14



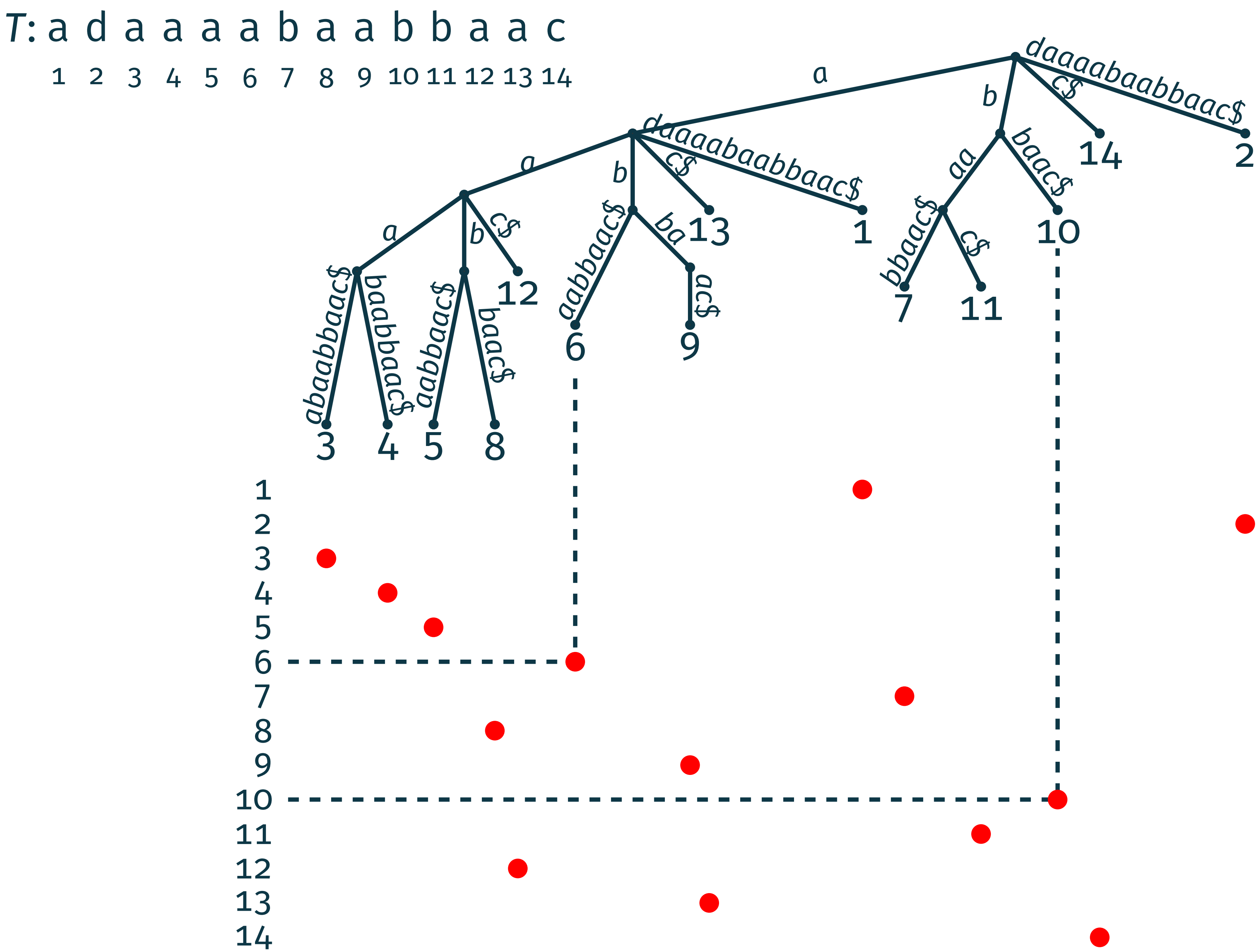
IPM queries $\rightarrow$ 2D Range Reporting

T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

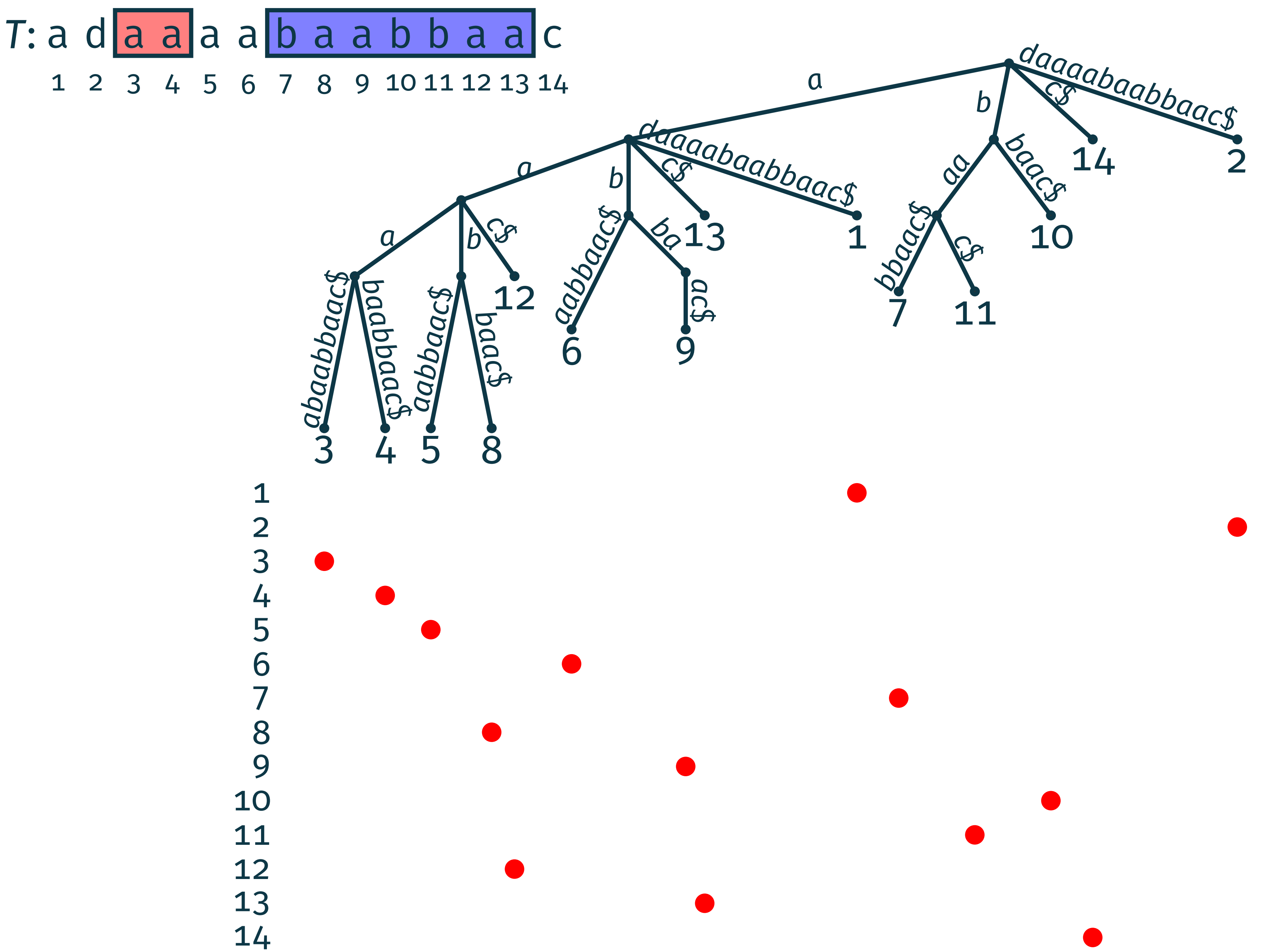


1
2
3
4
5
6
7
8
9
10
11
12
13
14

IPM queries → 2D Range Reporting



IPM queries → 2D Range Reporting

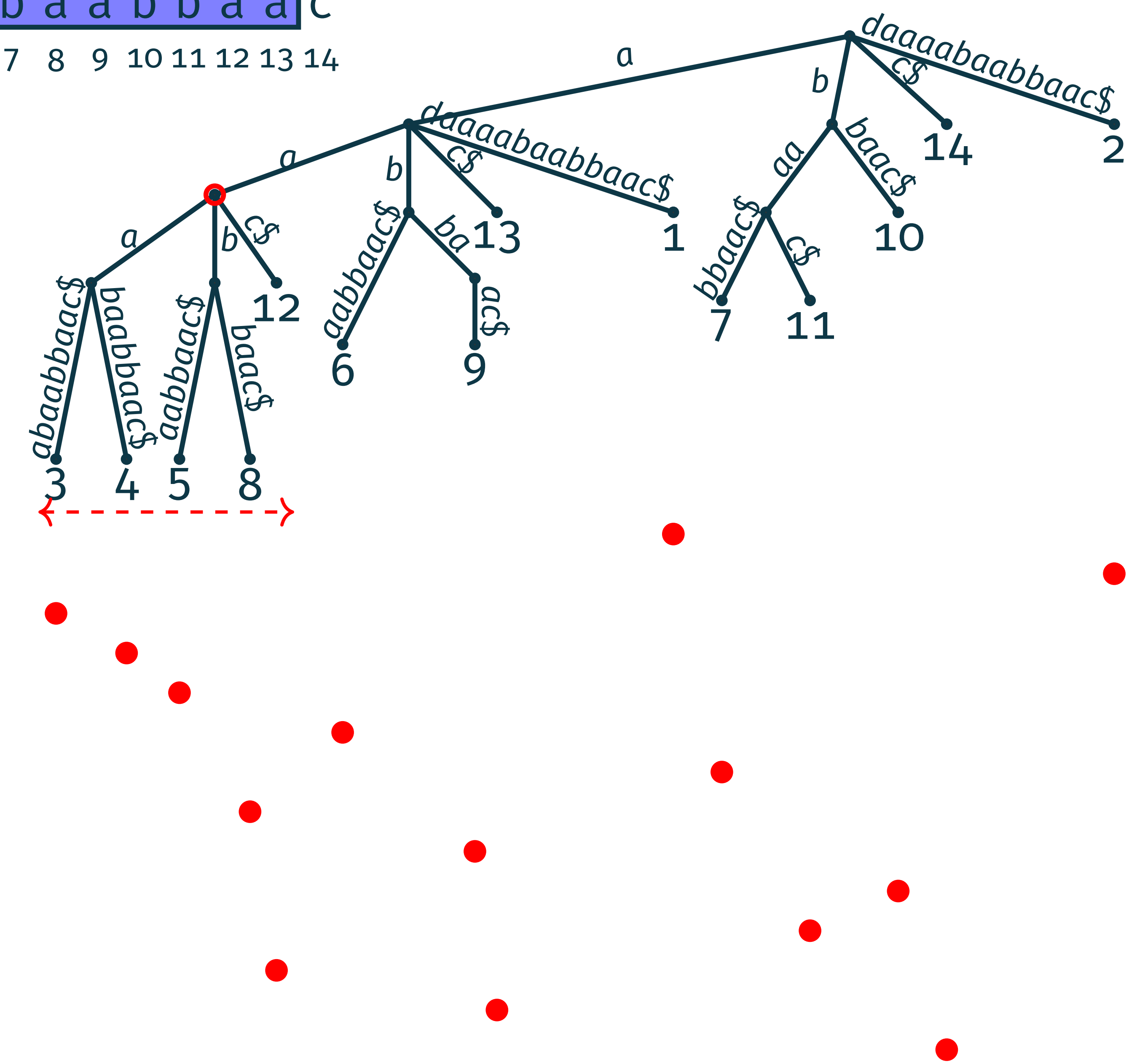


IPM queries $\rightarrow$ 2D Range Reporting

T: a d a a a a b a a b b a a c

1 2 3 4 5 6 7 8 9 10 11 12 13 14

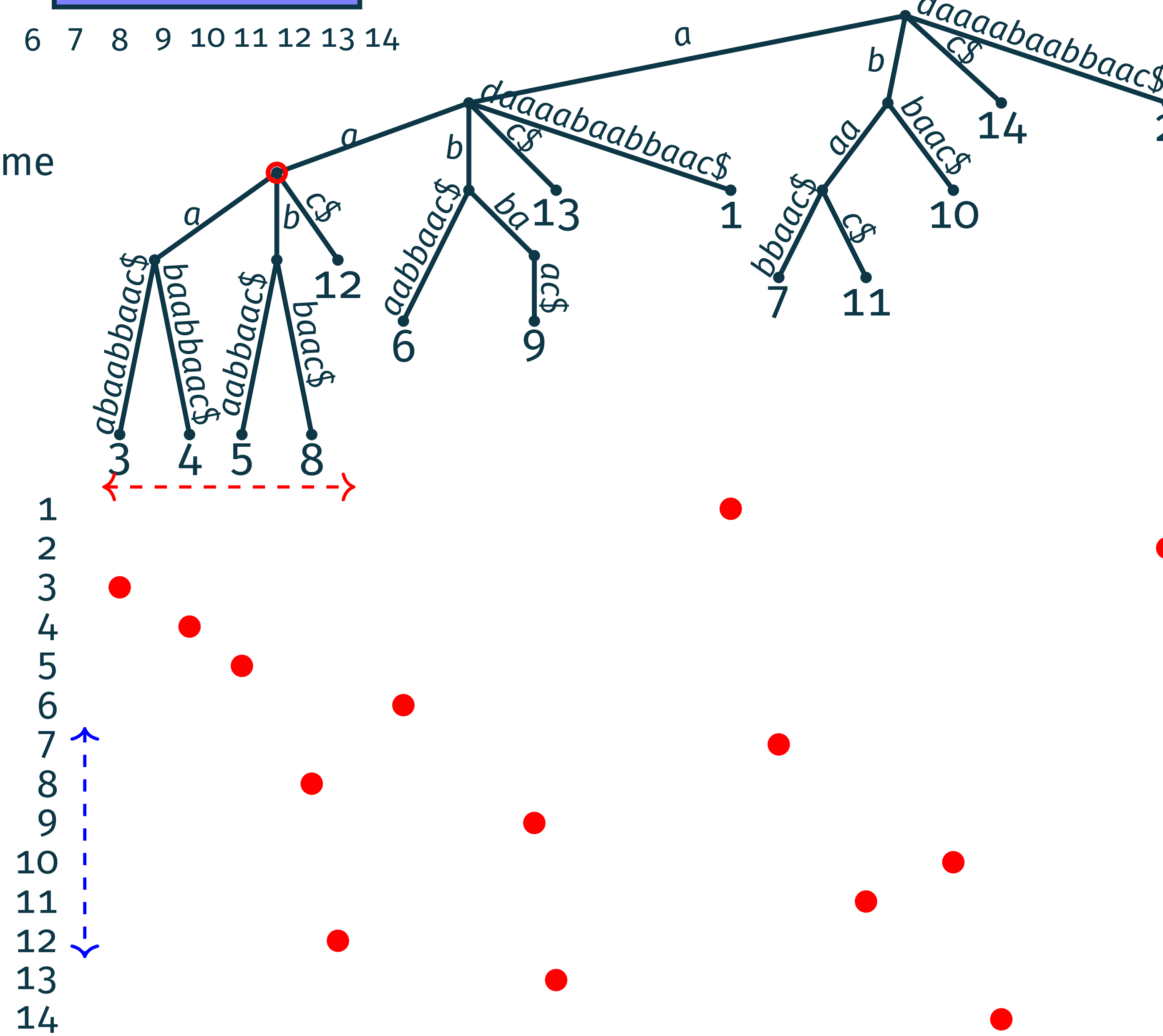
$\mathcal{O}(\log \log n)$ time



IPM queries $\rightarrow$ 2D Range Reporting

T:	a	d	a	a	a	a	b	a	a	b	b	a	a	c
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

$\mathcal{O}(\log \log n)$ time

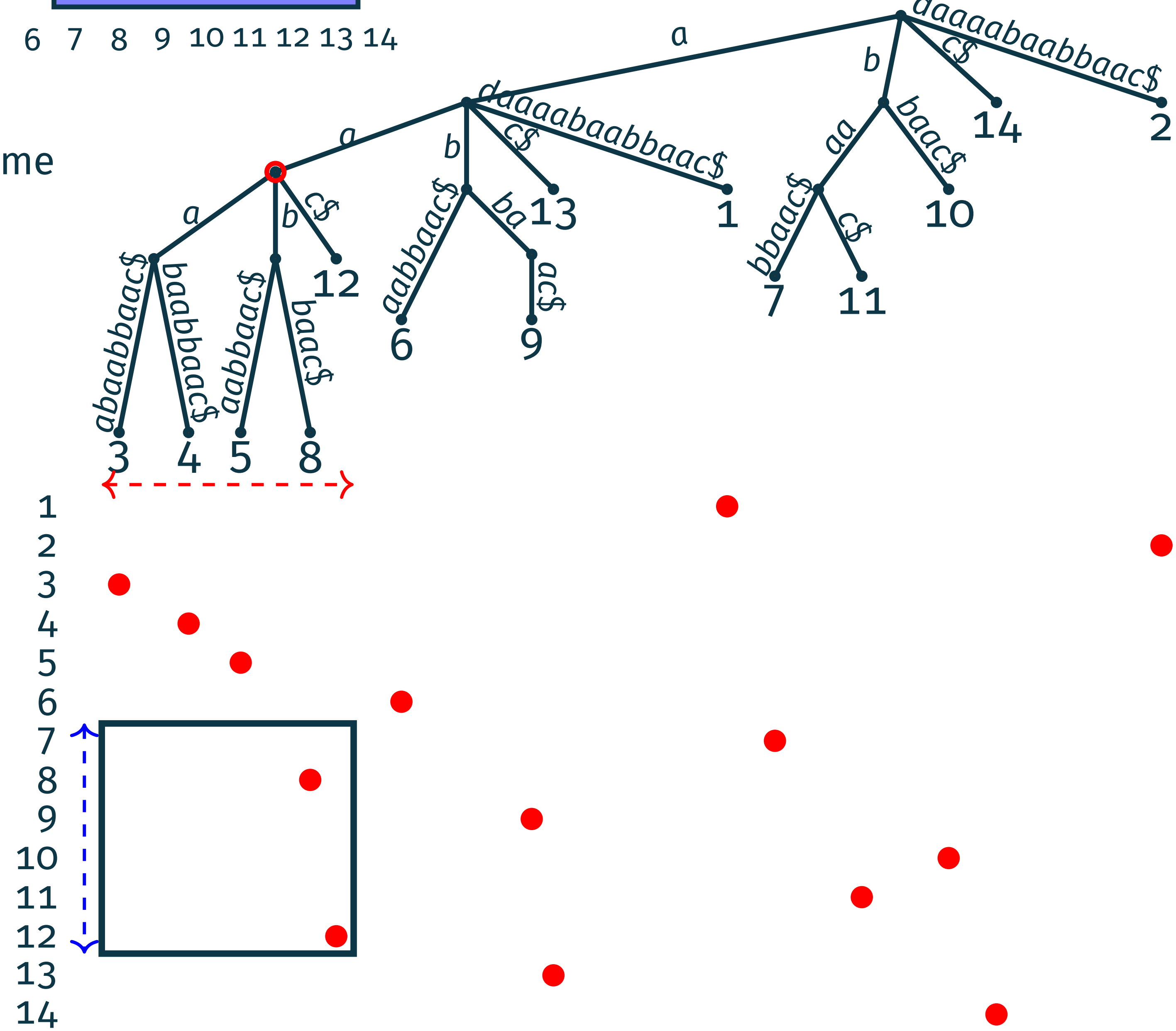


IPM queries → 2D Range Reporting

T: a d a a a a b a a b b a a c

1 2 3 4 5 6 7 8 9 10 11 12 13 14

$\mathcal{O}(\log \log n)$ time



Exercises

Exercises

Exercise: Given a fragment $S[i\,..\,j] = \texttt{abcabc}$ of a text and the period of this fragment, explain how we can find how much the periodicity extends (on both sides) using LCE queries. In other words, the task is to compute the longest fragment of S that contains $S[i\,..\,j]$ and has period 3.

Exercises

Exercise: Given a fragment $S[i\,..\,j] = \texttt{abcabc}$ of a text and the period of this fragment, explain how we can find how much the periodicity extends (on both sides) using LCE queries. In other words, the task is to compute the longest fragment of S that contains $S[i\,..\,j]$ and has period 3.

Exercise: Reduce a query that checks if a fragment of a string S is periodic (and if so, also returns its period) to an internal pattern matching query and a longest common extension query on S.

SLPs: Equal Fragments can be Parsed Differently

SLPs: Equal Fragments can be Parsed Differently

$X_0 \to \mathrm{b}$

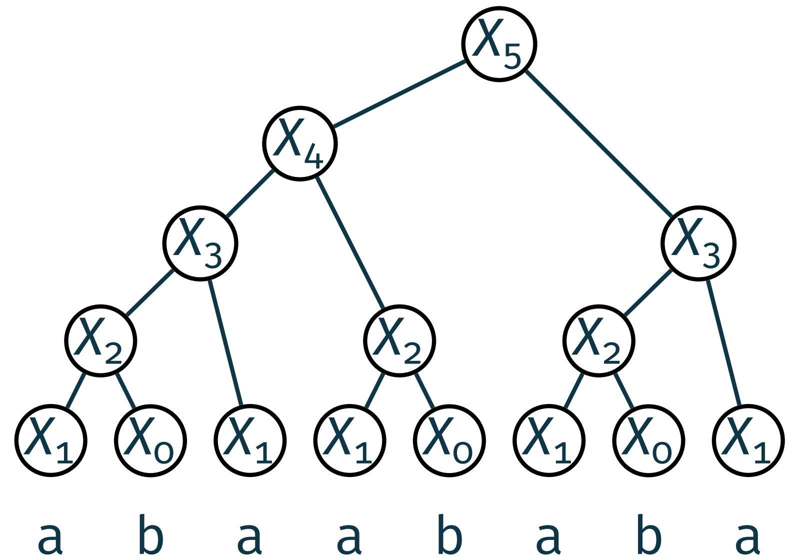
$X_1 \to \mathrm{a}$

$X_2 \to X_1 X_0$

$X_3 \to X_2 X_1$

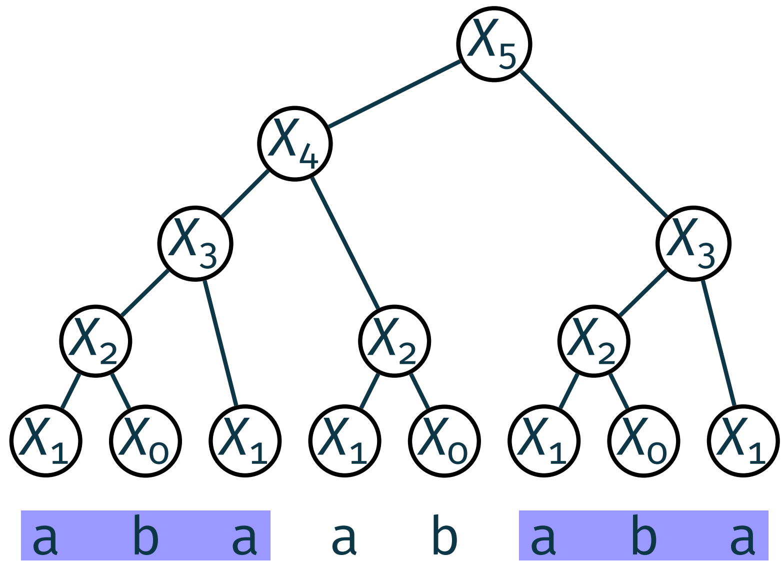
$X_4 \to X_3 X_2$

$X_5 \to X_4 X_3$



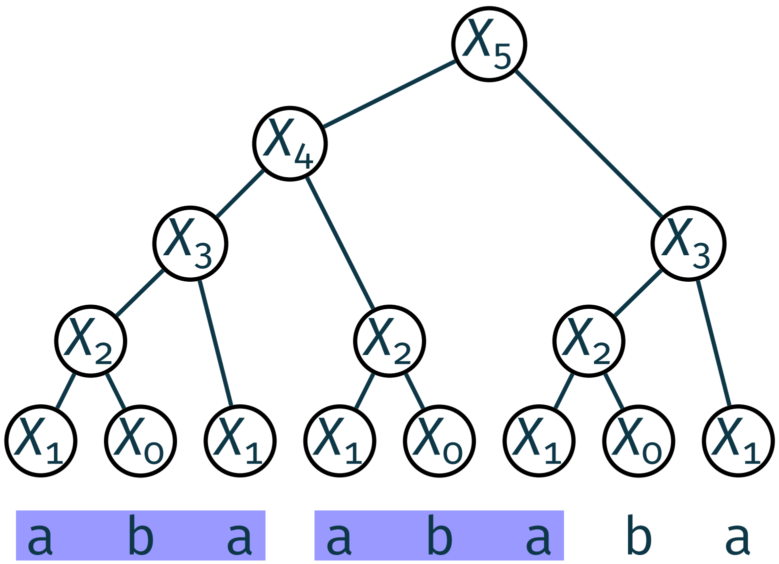
SLPs: Equal Fragments can be Parsed Differently

$$X_0 \to \mathrm{b}$$
$$X_1 \to \mathrm{a}$$
$$X_2 \to X_1X_0$$
$$X_3 \to X_2X_1$$
$$X_4 \to X_3X_2$$
$$X_5 \to X_4X_3$$



SLPs: Equal Fragments can be Parsed Differently

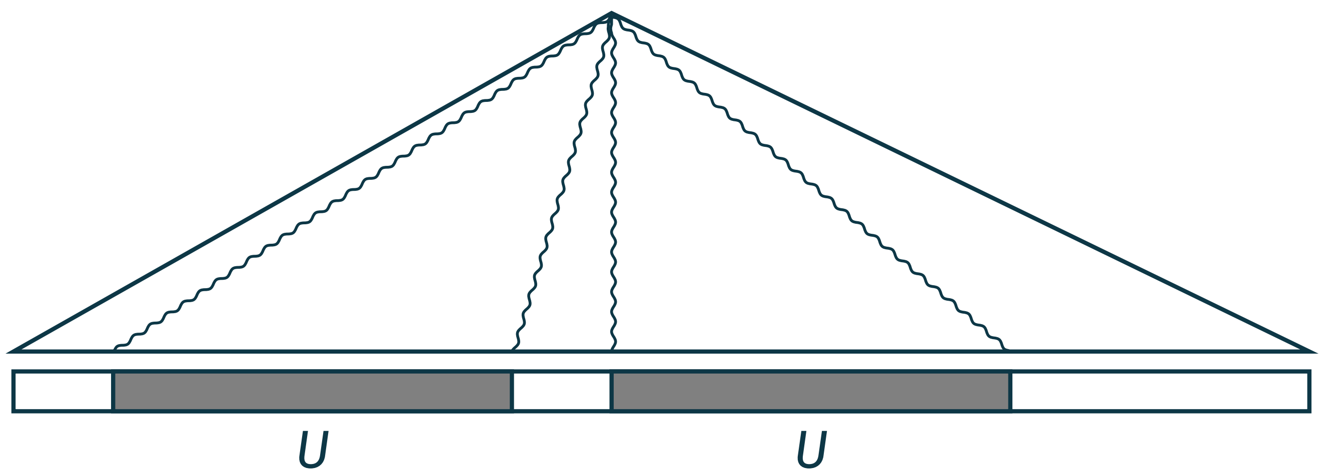
$$X_0 \to \mathrm{b}$$
$$X_1 \to \mathrm{a}$$
$$X_2 \to X_1X_0$$
$$X_3 \to X_2X_1$$
$$X_4 \to X_3X_2$$
$$X_5 \to X_4X_3$$



Locally Consistent Parsing

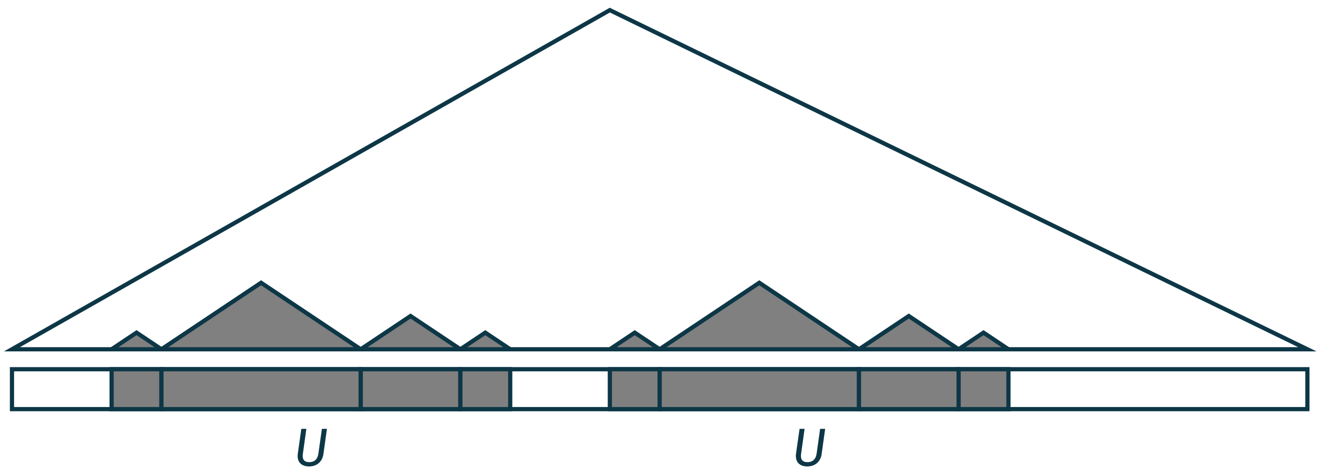
Locally Consistent Parsing

Locally Consistent Parsing: equal fragments are parsed similarly.



Locally Consistent Parsing

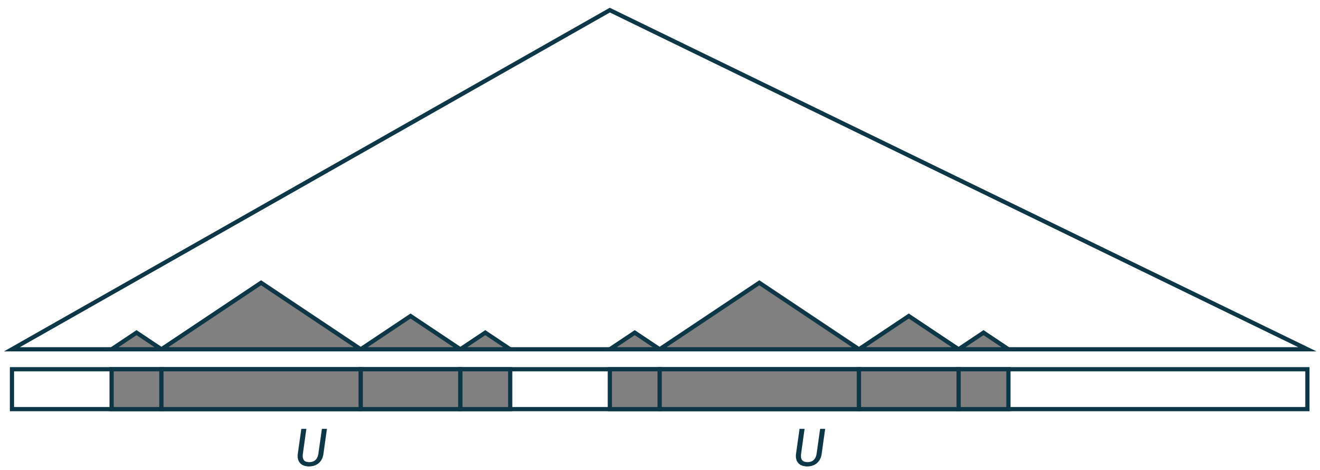
Locally Consistent Parsing: equal fragments are parsed similarly.



- Some nodes in the parse tree of w are preserved no matter the context.

Locally Consistent Parsing

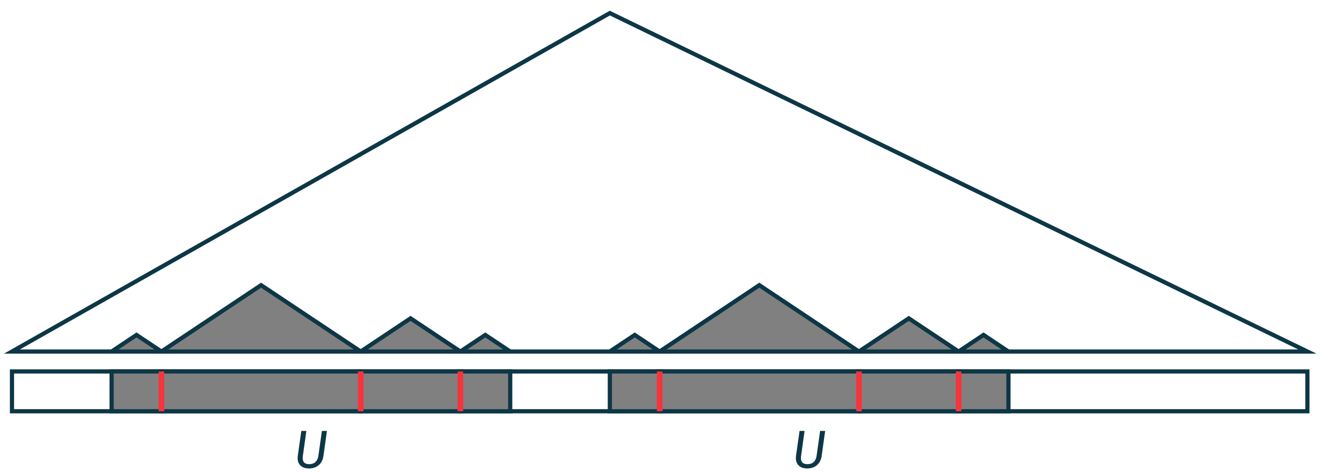
Locally Consistent Parsing: equal fragments are parsed similarly.



- Some nodes in the parse tree of w are preserved no matter the context.
- Topmost such nodes form a small layer: $\mathcal{O}(\log N)$ nodes.

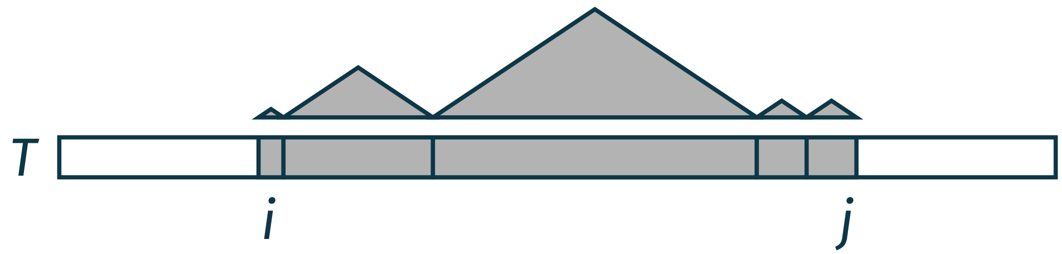
Locally Consistent Parsing

Locally Consistent Parsing: equal fragments are parsed similarly.

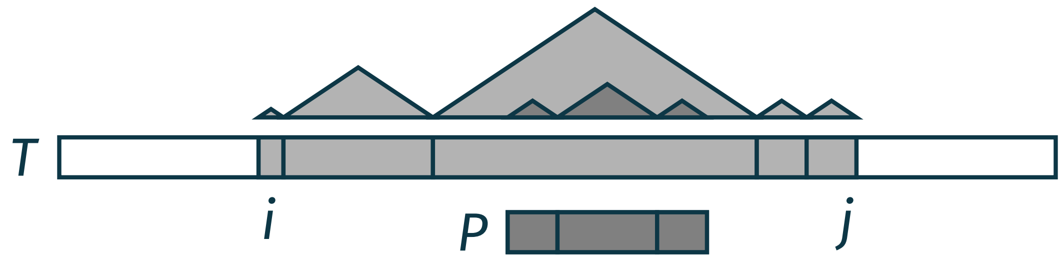


- Some nodes in the parse tree of w are preserved no matter the context.
- Topmost such nodes form a small layer: $\mathcal{O}(\log N)$ nodes.
- These nodes define $\mathcal{O}(\log N)$ breakpoints for each substring; they partition it into phrases.

Occurrences of P in T

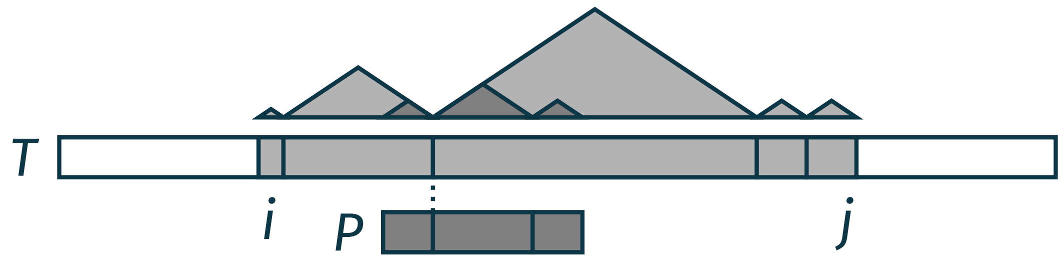


Occurrences of P in T



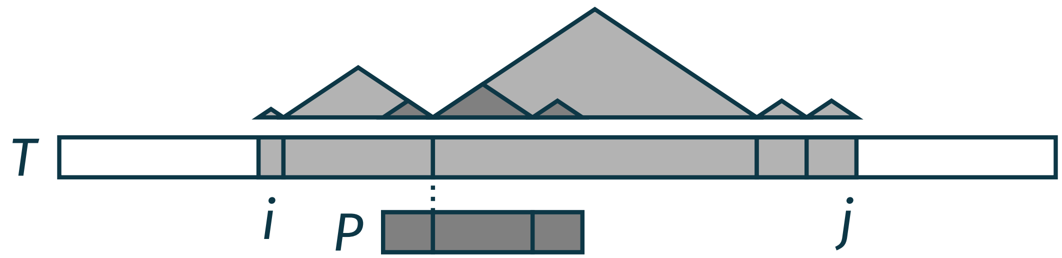
Either fully contained in a phrase

Occurrences of P in T



Either fully contained in a phrase, or a breakpoint of P is aligned with a breakpoint of $T[\,i\,..\,j\,]$.

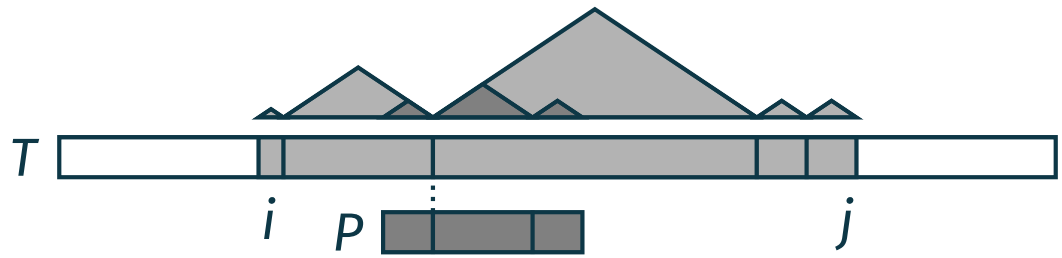
Occurrences of P in T



Either fully contained in a phrase, or a breakpoint of P is aligned with a breakpoint of $T[i\,..\,j]$.

Two SLPs that correspond to P and T, can be efficiently recompressed, so that the resulting parsings are locally consistent and have depth $\mathcal{O}(\log N)$.
[Jeż, TALG'15; Jeż, JACM'16; I, CPM'17]

Occurrences of P in T

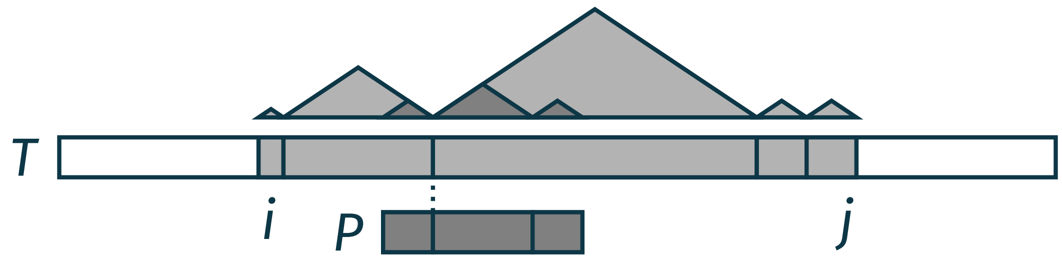


Either fully contained in a phrase, or a breakpoint of P is aligned with a breakpoint of $T[i\mathinner{.\,.}j]$.

Two SLPs that correspond to P and T, can be efficiently recompressed, so that the resulting parsings are locally consistent and have depth $\mathcal{O}(\log N)$.
[Jeż, TALG'15; Jeż, JACM'16; I, CPM'17]

How can we efficiently compute the occurrences of a substring U in a substring V if $|V| < 2|U|$?

Occurrences of P in T



Either fully contained in a phrase, or a breakpoint of P is aligned with a breakpoint of $T[\,i\mathinner{.\,.}j\,]$.

Two SLPs that correspond to P and T, can be efficiently recompressed, so that the resulting parsings are locally consistent and have depth $\mathcal{O}(\log N)$.
[Jeż, TALG'15; Jeż, JACM'16; I, CPM'17]

How can we efficiently compute the occurrences of a substring U in a substring V if $|V| < 2|U|$? For each non-terminal in the parse tree whose production "breaks" a fragment of V of length at least m, try to align the breakpoint with each of U's $\mathcal{O}(\log N)$ breakpoints!

Recompression

Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

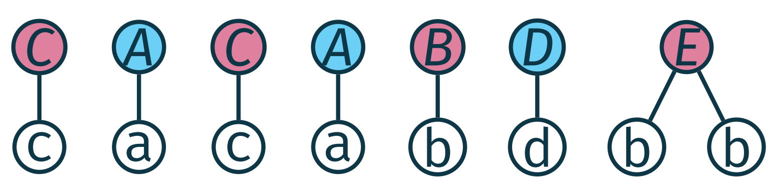
ⓒ ⓐ ⓒ ⓐ ⓑ ⓓ ⓑ ⓑ

Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **RunCompress** – for each B^r, $r > 1$, replace all occurrences of B^r as a run by a new letter A.



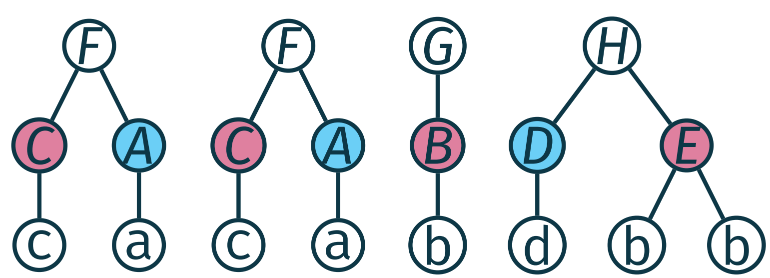
RunCompress

Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **HalfCompress** – partition Σ into Σ_ℓ and Σ_r and replace all occurrences of BC ($B \in \Sigma_\ell$ and $C \in \Sigma_r$) by a new letter A.



HalfCompress

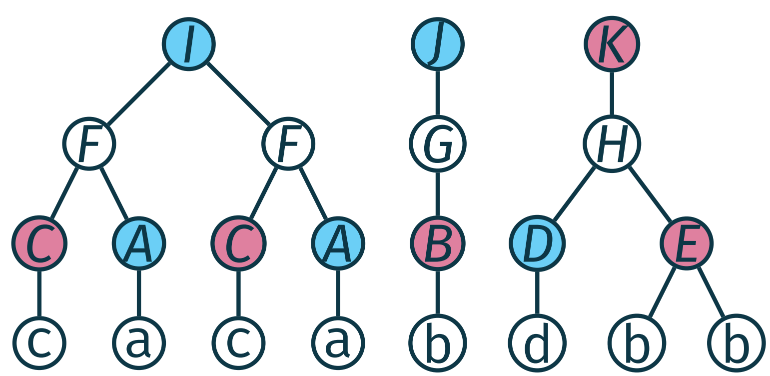
RunCompress

Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **RunCompress** – for each B^r, $r > 1$, replace all occurrences of B^r as a run by a new letter A.



RunCompress

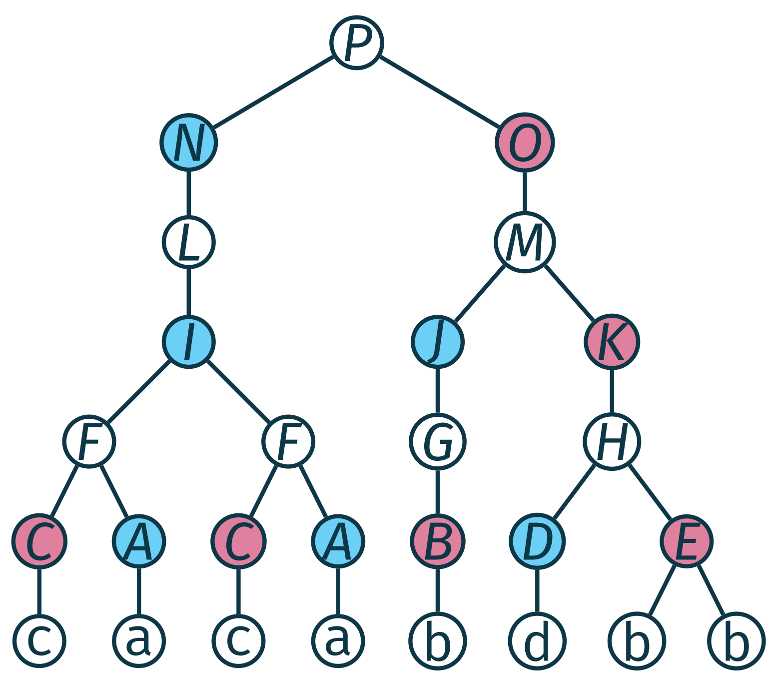
HalfCompress

RunCompress

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **HalfCompress** – partition Σ into Σ_ℓ and Σ_r and replace all occurrences of BC ($B \in \Sigma_\ell$ and $C \in \Sigma_r$) by a new letter A.



HalfCompress

RunCompress

HalfCompress

RunCompress

HalfCompress

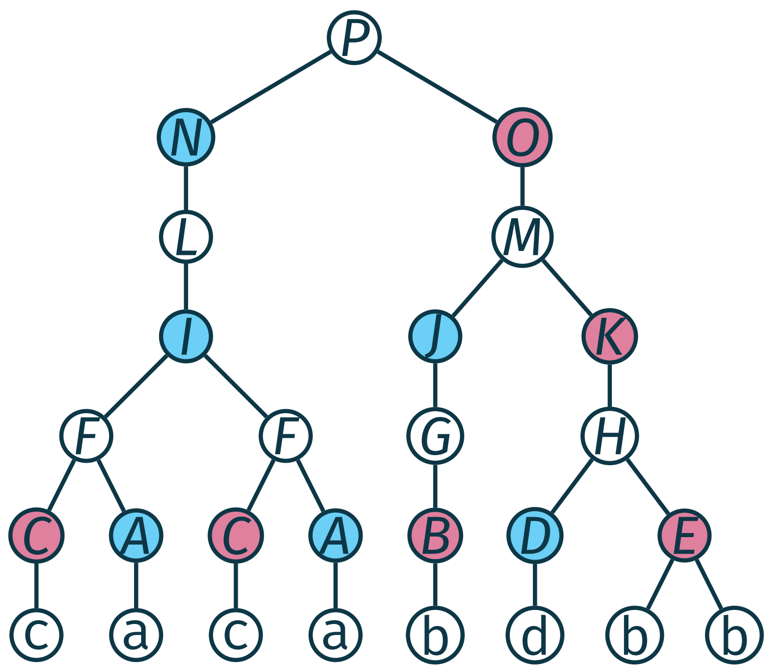
RunCompress

Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **RunCompress** – for each B^r, $r > 1$, replace all occurrences of B^r as a run by a new letter A.
- **HalfCompress** – partition Σ into Σ_ℓ and Σ_r and replace all occurrences of BC ($B \in \Sigma_\ell$ and $C \in \Sigma_r$) by a new letter A.



HalfCompress

RunCompress

HalfCompress

RunCompress

HalfCompress

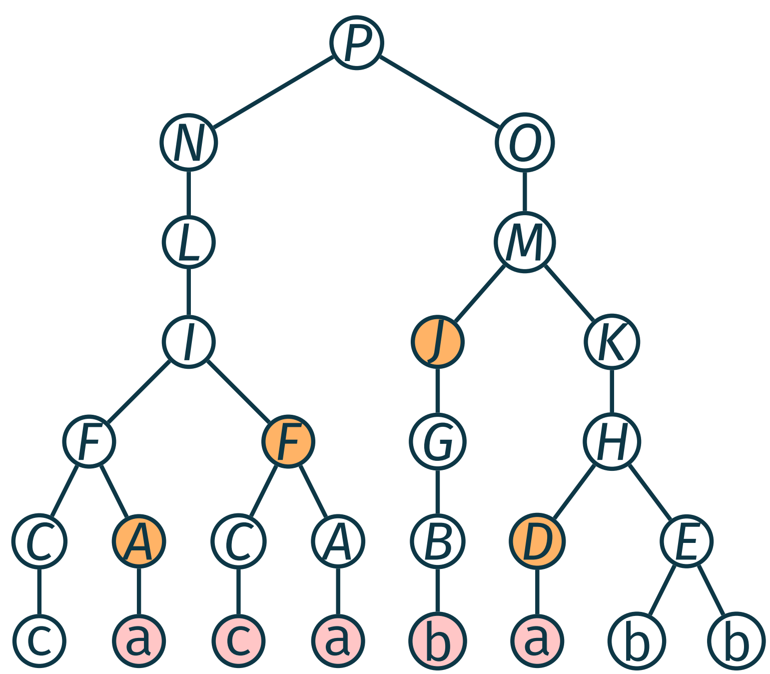
RunCompress

Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **RunCompress** – for each B^r, $r > 1$, replace all occurrences of B^r as a run by a new letter A.
- **HalfCompress** – partition Σ into Σ_ℓ and Σ_r and replace all occurrences of BC ($B \in \Sigma_\ell$ and $C \in \Sigma_r$) by a new letter A.

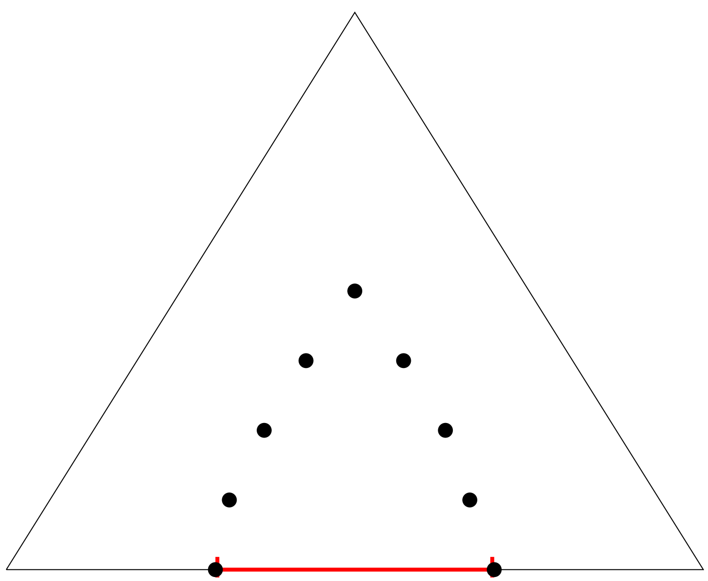


Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **RunCompress** – for each B^r, $r > 1$, replace all occurrences of B^r as a run by a new letter A.
- **HalfCompress** – partition Σ into Σ_ℓ and Σ_r and replace all occurrences of BC ($B \in \Sigma_\ell$ and $C \in \Sigma_r$) by a new letter A.

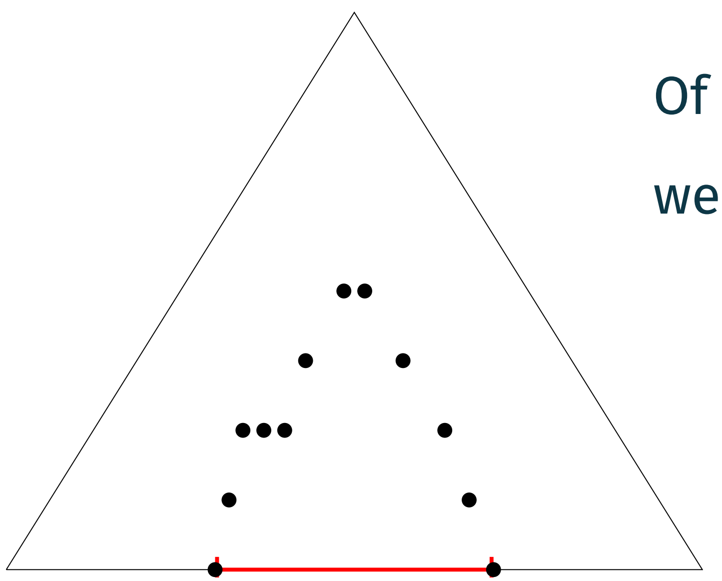


Recompression

Run-length straight line program: context-free grammar generating exactly one string T by concatenations $A \to BC$ and powers $A \to B^k$.

Perform two types of steps interleaved until $|T| = 1$:

- **RunCompress** – for each B^r, $r > 1$, replace all occurrences of B^r as a run by a new letter A.
- **HalfCompress** – partition Σ into Σ_ℓ and Σ_r and replace all occurrences of BC ($B \in \Sigma_\ell$ and $C \in \Sigma_r$) by a new letter A.



Of course, periodicity causes trouble:

we have $\mathcal{O}(\log N)$ groups of breakpoints.

PILLAR Operations in the Compressed Setting

PILLAR Operations in the Compressed Setting

After a preprocessing that runs in time nearly-linear in the size of the SLPs that generate P and T, we can perform each PILLAR operation in $\mathcal{O}(\log^2 N \log\log N)$ time.

[I, CPM'17; CKW'20]

Approximate Pattern Matching under Edit Distance: Summary of Results

Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$ [Sellers; J. Algorithms 1980]

Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$ [Sellers; J. Algorithms 1980]

$\mathcal{O}(nk^2)$ [Landau, Vishkin; JCSS 1988]

Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$	[Sellers; J. Algorithms 1980]
$\mathcal{O}(nk^2)$	[Landau, Vishkin; JCSS 1988]
$\mathcal{O}(nk)$	[Landau, Vishkin; J. Algorithms 1989]

Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$	[Sellers; J. Algorithms 1980]
$\mathcal{O}(nk^2)$	[Landau, Vishkin; JCSS 1988]
$\mathcal{O}(nk)$	[Landau, Vishkin; J. Algorithms 1989]
$\tilde{\mathcal{O}}(n + k^{8+1/3} \cdot n/m^{1/3})$	[Sahinalp, Vishkin; FOCS 1996]

Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$	[Sellers; J. Algorithms 1980]
$\mathcal{O}(nk^2)$	[Landau, Vishkin; JCSS 1988]
$\mathcal{O}(nk)$	[Landau, Vishkin; J. Algorithms 1989]
$\tilde{\mathcal{O}}(n + k^{8+1/3} \cdot n/m^{1/3})$	[Sahinalp, Vishkin; FOCS 1996]
$\mathcal{O}(n + k^4 \cdot n/m)$	[Cole, Hariharan; SICOMP 2002]

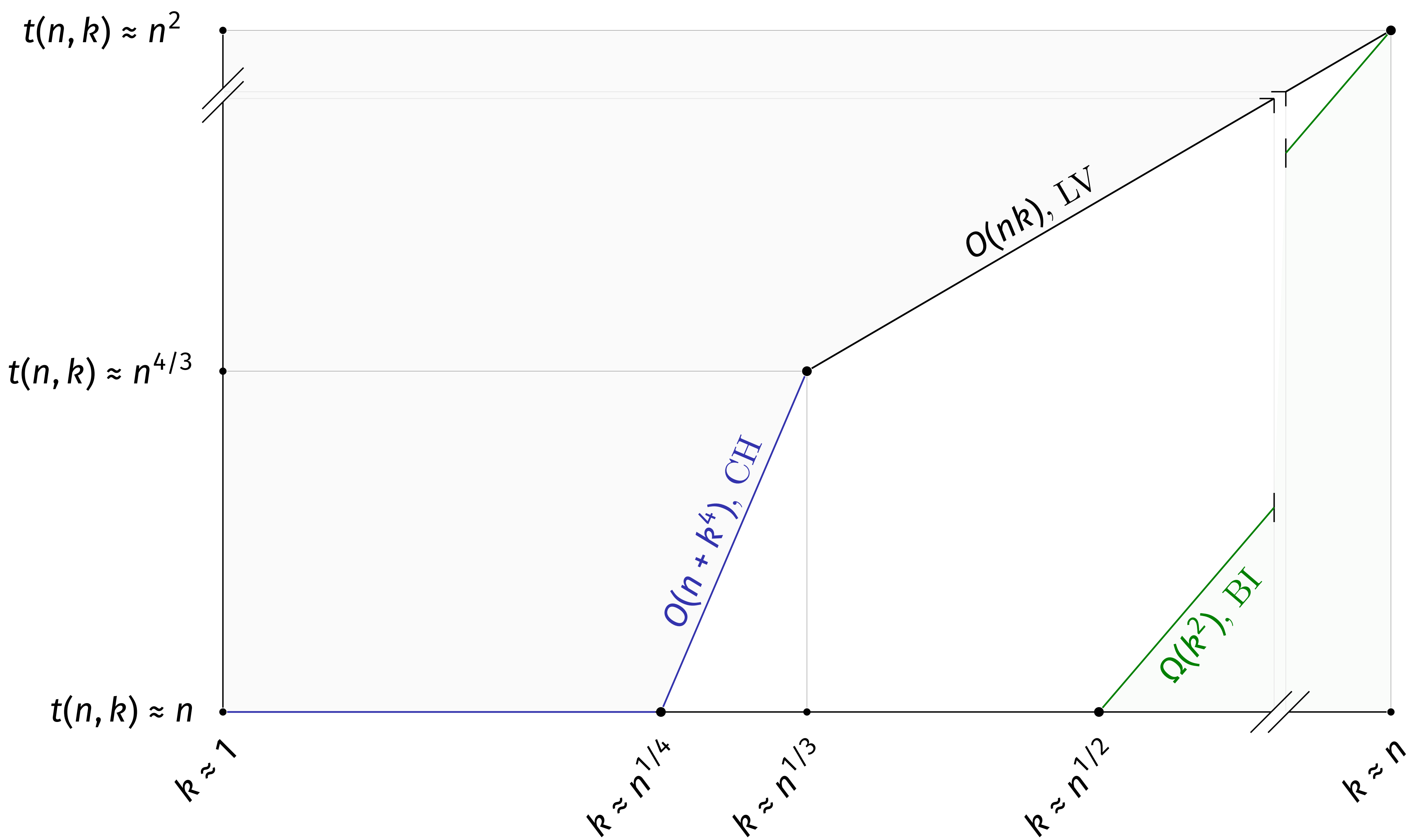
Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$	[Sellers; J. Algorithms 1980]
$\mathcal{O}(nk^2)$	[Landau, Vishkin; JCSS 1988]
$\mathcal{O}(nk)$	[Landau, Vishkin; J. Algorithms 1989]
$\tilde{\mathcal{O}}(n + k^{8+1/3} \cdot n/m^{1/3})$	[Sahinalp, Vishkin; FOCS 1996]
$\mathcal{O}(n + k^4 \cdot n/m)$	[Cole, Hariharan; SICOMP 2002]
$\tilde{\mathcal{O}}(n + k^{3.5} \cdot n/m)$	Using the structural result [CKW'22]

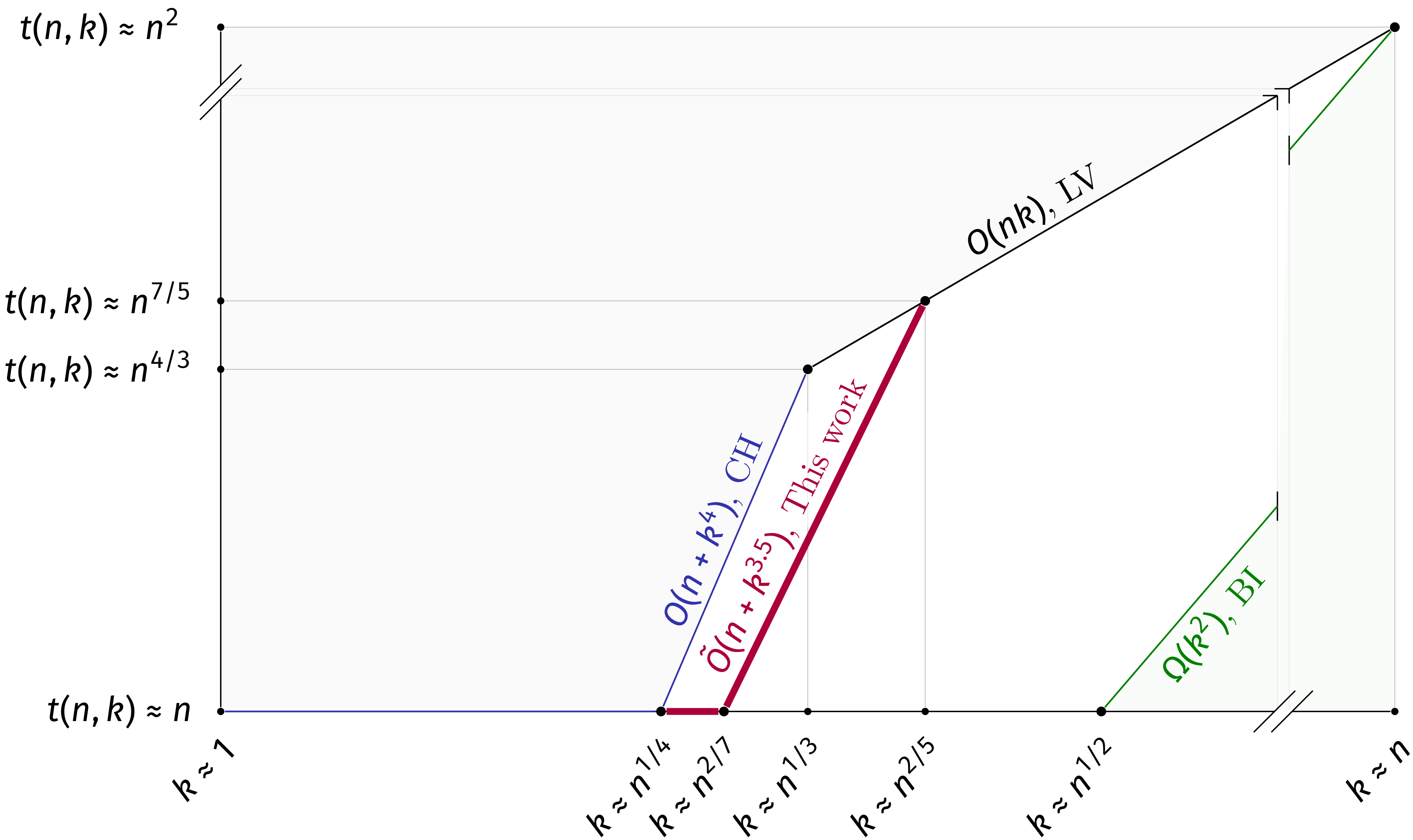
Approximate Pattern Matching under Edit Distance: Summary of Results

$\mathcal{O}(n^2)$	[Sellers; J. Algorithms 1980]
$\mathcal{O}(nk^2)$	[Landau, Vishkin; JCSS 1988]
$\mathcal{O}(nk)$	[Landau, Vishkin; J. Algorithms 1989]
$\tilde{\mathcal{O}}(n + k^{8+1/3} \cdot n/m^{1/3})$	[Sahinalp, Vishkin; FOCS 1996]
$\mathcal{O}(n + k^4 \cdot n/m)$	[Cole, Hariharan; SICOMP 2002]
$\tilde{\mathcal{O}}(n + k^{3.5} \cdot n/m)$	Using the structural result [CKW'22]
$\Omega(k^2)$	[Backurs, Indyk; SICOMP 2018]

Approximate Pattern Matching under Edit Distance: Summary of Results



Approximate Pattern Matching under Edit Distance: Summary of Results



The Structure of Pattern Matching under Edit Distance

The Structure of Pattern Matching under Edit Distance

Theorem [CKW; FOCS'20] Given a pattern P of length m and a text T of length $n \le {}^3\!/\!_2\, m$, and a threshold $k \le m$ at least one of the following holds:

The Structure of Pattern Matching under Edit Distance

Theorem [CKW; FOCS'20] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$, and a threshold $k \leq m$ at least one of the following holds:

- The pattern P has $\mathcal{O}(k^2)$ k-error occurrences in T.

The Structure of Pattern Matching under Edit Distance

Theorem [CKW; FOCS'20] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$, and a threshold $k \leq m$ at least one of the following holds:

- The pattern P has $\mathcal{O}(k^2)$ k-error occurrences in T.
- The pattern is almost periodic: at edit distance $< 2k$ from a string with period $\mathcal{O}(m/k)$.

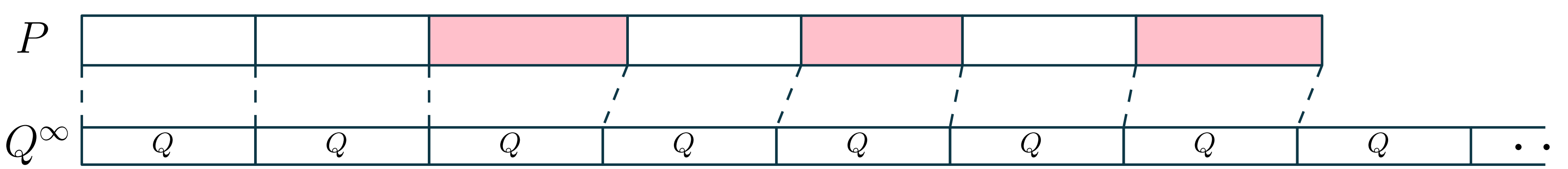
The Structure of Pattern Matching under Edit Distance

Theorem [CKW; FOCS'20] Given a pattern P of length m and a text T of length $n \leq {}^{3}\!/_{2}\, m$, and a threshold $k \leq m$ at least one of the following holds:

- The pattern P has $\mathcal{O}(k^2)$ k-error occurrences in T.
- The pattern is almost periodic: at edit distance $< 2k$ from a string with period $\mathcal{O}(m/k)$. **This is the bottleneck.**

Theorem [CKW; FOCS'20] Given a pattern P of length m and a text T of length $n \leq 3/2\, m$, and a threshold $k \leq m$ at least one of the following holds:

- The pattern P has $\mathcal{O}(k^2)$ k-error occurrences in T.
- The pattern is almost periodic: at edit distance $< 2k$ from a string with period $\mathcal{O}(m/k)$. **This is the bottleneck.**

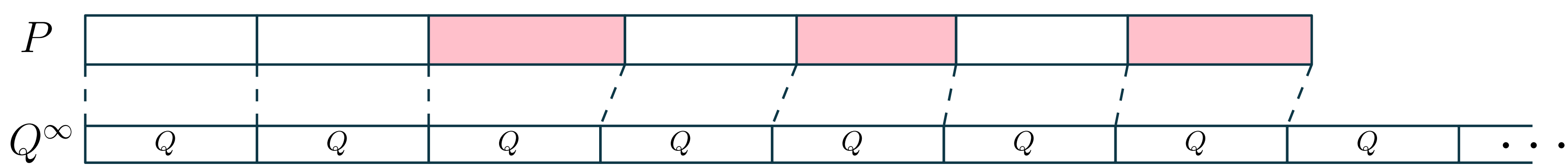


Q will denote a primitive string; it does not match any of its rotations.

The Structure of Pattern Matching under Edit Distance

Theorem [CKW; FOCS'20] Given a pattern P of length m and a text T of length $n \leq {}^3\!/_2\, m$, and a threshold $k \leq m$ at least one of the following holds:

- The pattern P has $\mathcal{O}(k^2)$ k-error occurrences in T.
- The pattern is almost periodic: at edit distance $< 2k$ from a string with period $\mathcal{O}(m/k)$. **This is the bottleneck.**



Q will denote a primitive string; it does not match any of its rotations.

We call this a tile decomposition of P with respect to Q.

Reduction to the Almost Periodic Case [CKW'20]

Reduction to the Almost Periodic Case [CKW'20]

$\mathcal{O}(k^4 \cdot n/m)$ `PILLAR`-time algorithm [CKW'20] matches [Cole, Hariharan; SICOMP 2002] for the standard setting.

Reduction to the Almost Periodic Case [CKW'20]

$\mathcal{O}(k^4 \cdot n/m)$ PILLAR-time algorithm [CKW'20] matches [Cole, Hariharan; SICOMP 2002] for the standard setting.

Reduction [CKW'20]: An algorithm that solves the almost periodic case in $\tilde{\mathcal{O}}(k^a \cdot n/m)$ PILLAR-time, for $a \geq 3$, implies an algorithm that solves the general case in $\tilde{\mathcal{O}}(k^a \cdot n/m)$ PILLAR-time.

Dynamic Puzzle Matching

Dynamic Puzzle Matching

Input: An integer k and a family $\mathcal{F}$ of strings containing a distinguished primitive string Q with $\sum_{F \in \mathcal{F}} \delta_E(F, Q) = \mathcal{O}(k)$.

Dynamic Puzzle Matching

Input: An integer k and a family $\mathcal{F}$ of strings containing a distinguished primitive string Q with $\sum_{F \in \mathcal{F}} \delta_E(F, Q) = \mathcal{O}(k)$.

Maintain: A sequence $\mathcal{I} = (U_1, V_1) \cdots (U_z, V_z)$ of pairs from $\mathcal{F}^2$.

Dynamic Puzzle Matching

Input: An integer k and a family $\mathcal{F}$ of strings containing a distinguished primitive string Q with $\sum_{F \in \mathcal{F}} \delta_E(F, Q) = \mathcal{O}(k)$.

Maintain: A sequence $\mathcal{I} = (U_1, V_1) \cdots (U_z, V_z)$ of pairs from $\mathcal{F}^2$.

Updates: Insertions and deletions of pairs in $\mathcal{I}$.

Dynamic Puzzle Matching

Input: An integer k and a family $\mathcal{F}$ of strings containing a distinguished primitive string Q with $\sum_{F \in \mathcal{F}} \delta_E(F, Q) = \mathcal{O}(k)$.

Maintain: A sequence $\mathcal{I} = (U_1, V_1) \cdots (U_z, V_z)$ of pairs from $\mathcal{F}^2$.

Updates: Insertions and deletions of pairs in $\mathcal{I}$.

Queries: Compute the k-error occurrences of $U_1 \cdots U_z$ in $V_1 \cdots V_z$.

Dynamic Puzzle Matching

Input: An integer k and a family $\mathcal{F}$ of strings containing a distinguished primitive string Q with $\sum_{F\in\mathcal{F}} \delta_E(F, Q) = \mathcal{O}(k)$.

Maintain: A sequence $\mathcal{I} = (U_1, V_1) \cdots (U_z, V_z)$ of pairs from $\mathcal{F}^2$.

Updates: Insertions and deletions of pairs in $\mathcal{I}$.

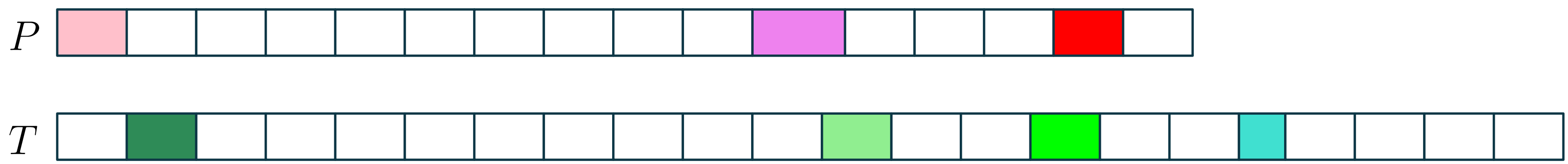
Queries: Compute the k-error occurrences of $U_1 \cdots U_z$ in $V_1 \cdots V_z$.

After $\tilde{\mathcal{O}}(k^3)$-time preprocessing, updates and queries take $\tilde{\mathcal{O}}(k)$ time.

Using Dynamic Puzzle Matching

Using Dynamic Puzzle Matching

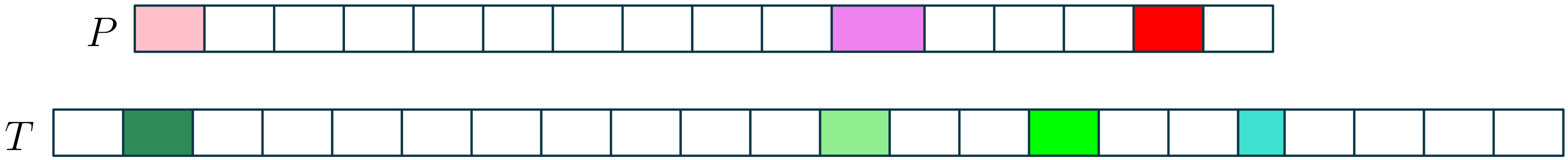
Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



Each string has $\mathcal{O}(k)$ special tiles.

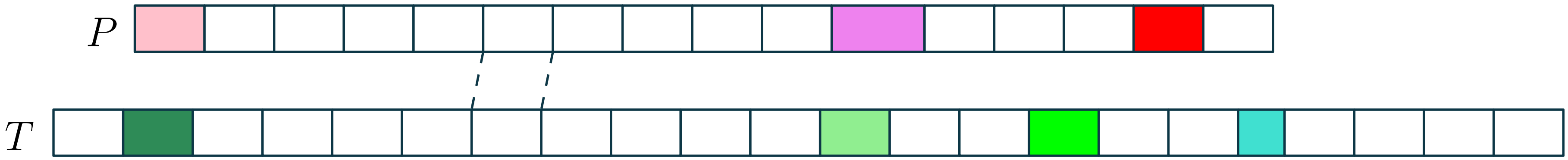
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



Using Dynamic Puzzle Matching

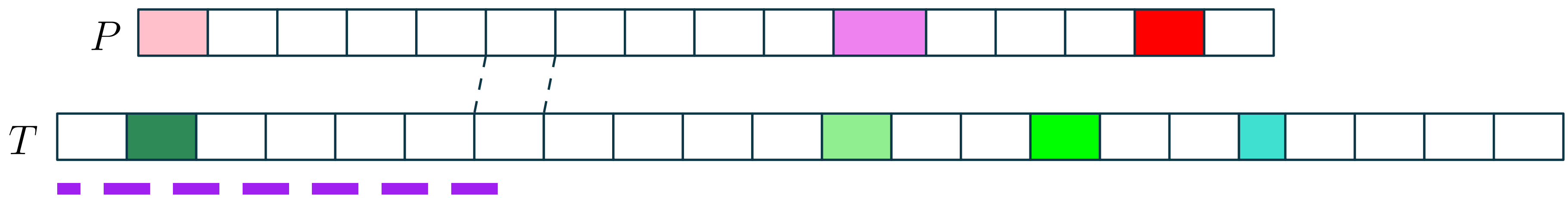
Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



$>k$ copies of Q in $P \implies \geq 1$ must be matched exactly

Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.

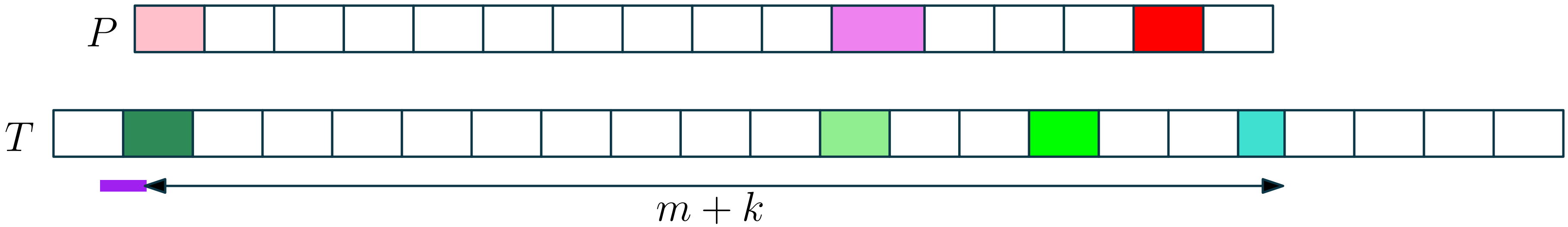


$> k$ copies of Q in $P \implies \geq 1$ must be matched exactly

Starting positions of k-error occs in T are within $\mathcal{O}(k)$ from endpoints of tiles.

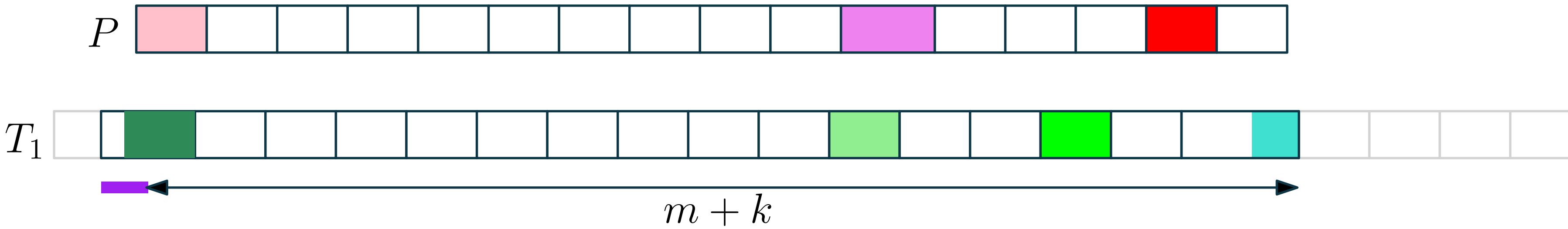
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



Using Dynamic Puzzle Matching

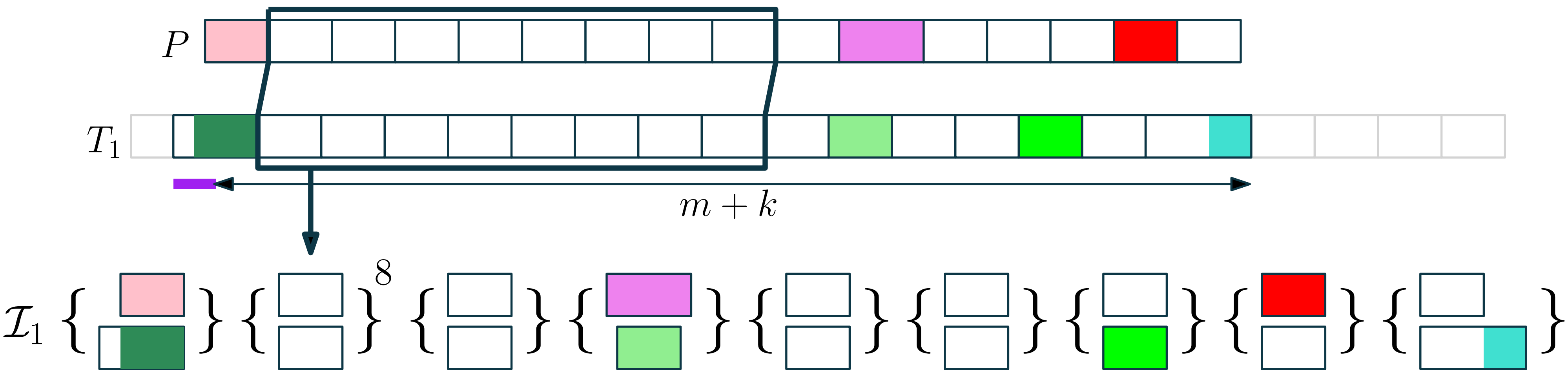
Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



$$|T_j| = m + \mathcal{O}(k)$$

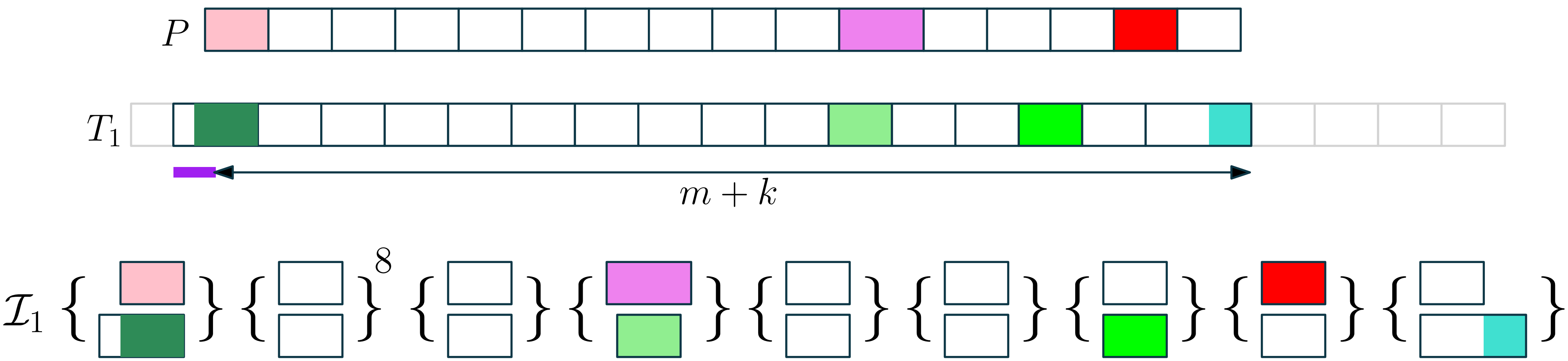
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



Using Dynamic Puzzle Matching

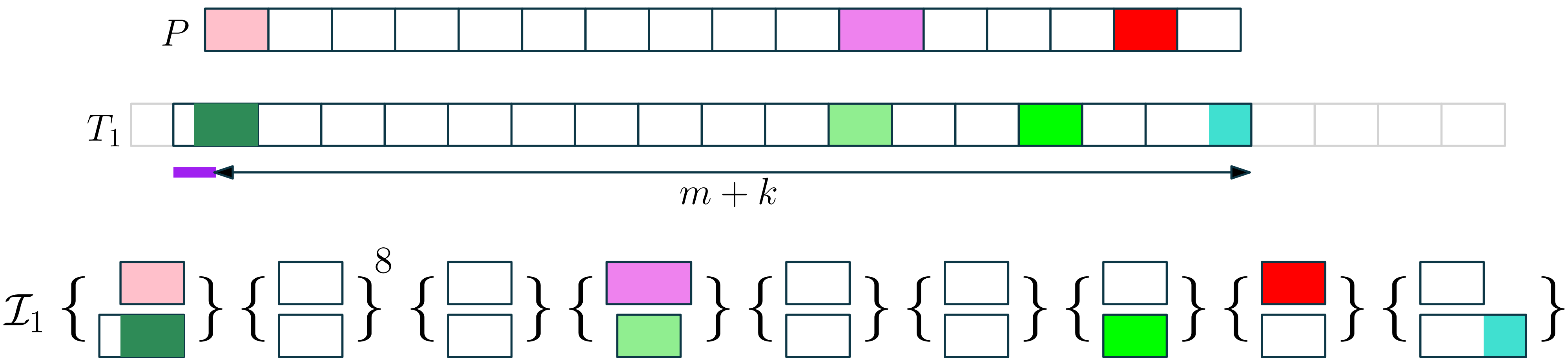
Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



Goal: Iterate over all $\mathcal{I}_j$'s in a DPM instance.

Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.

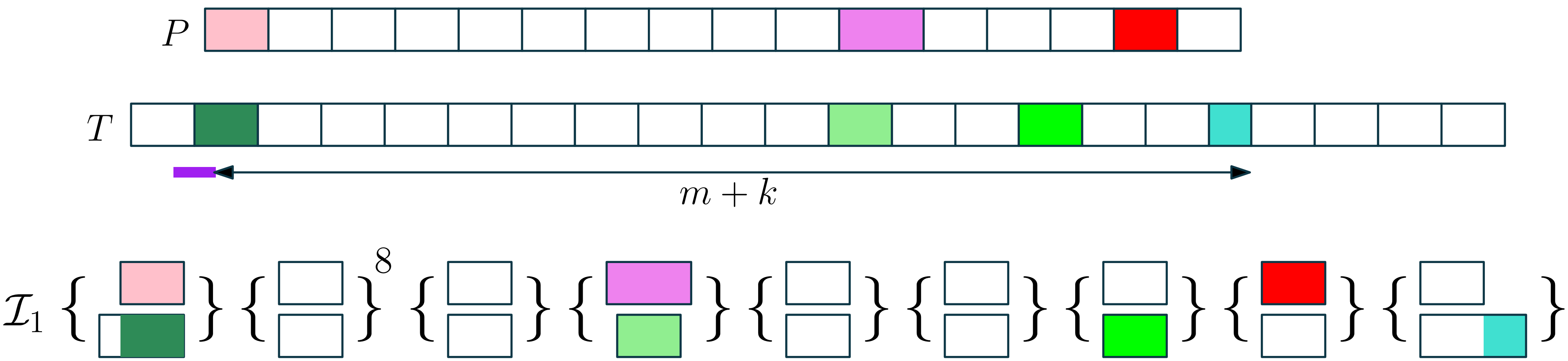


Goal: Iterate over all $\mathcal{I}_j$'s in a DPM instance.

(The leading and trailing pairs are treated separately.)

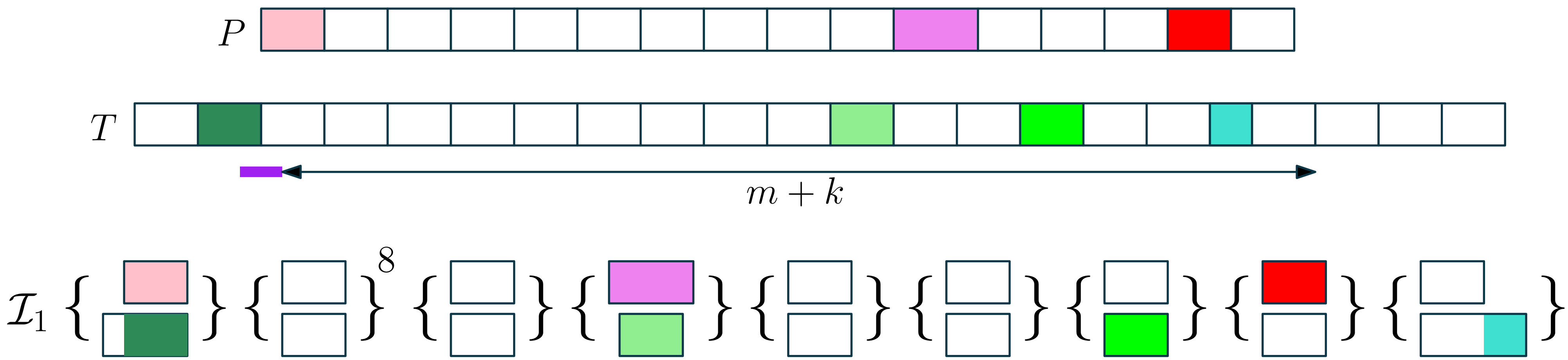
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



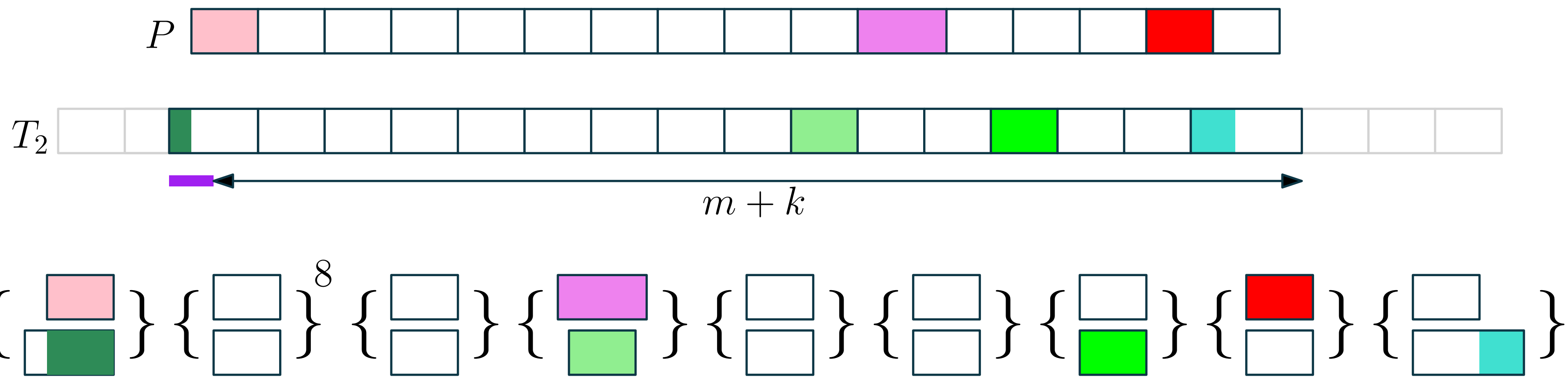
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



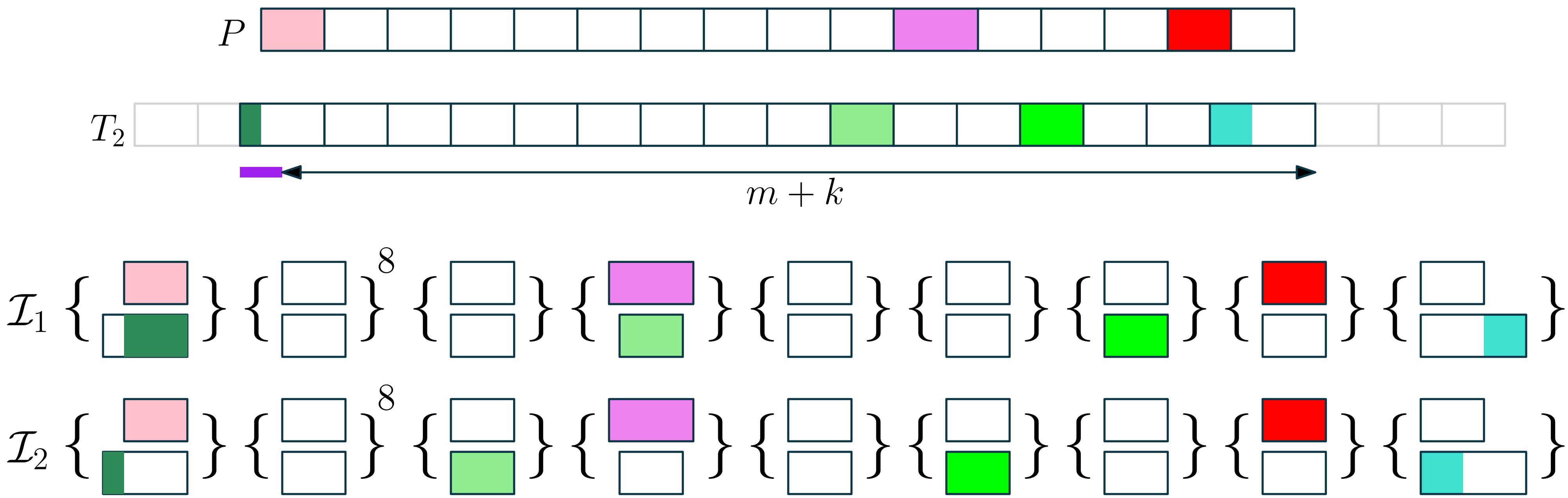
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



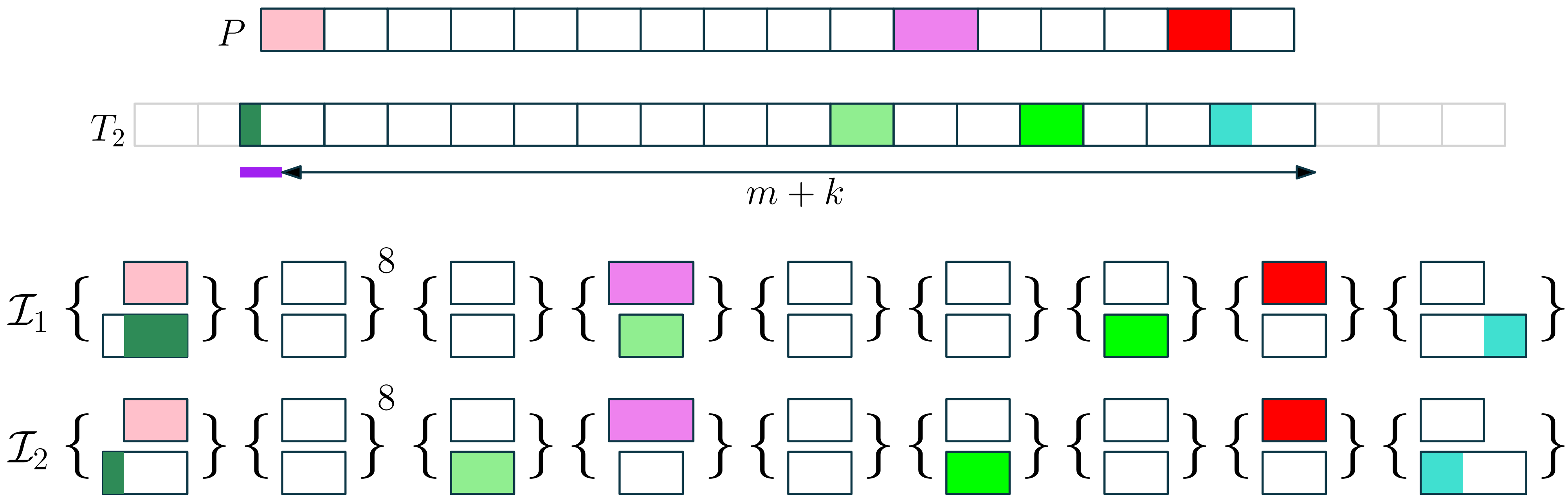
Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



Using Dynamic Puzzle Matching

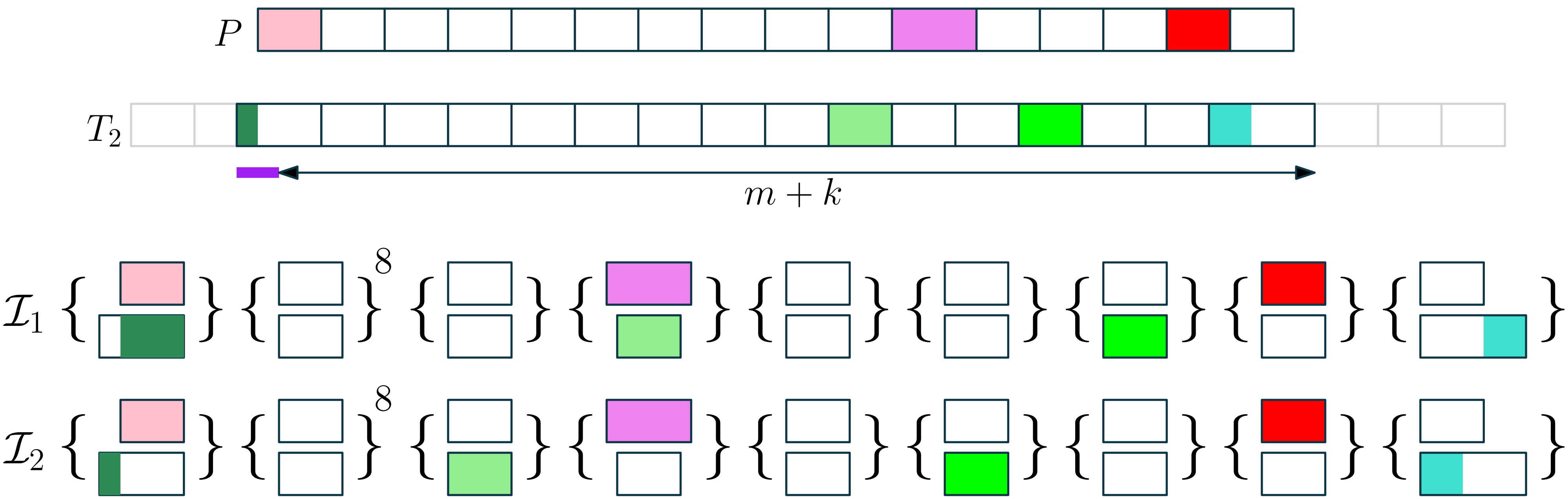
Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



We only need to update $\mathcal{O}(k)$ pairs; there has to be a pair $\neq (Q, Q)$ involved!

Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.

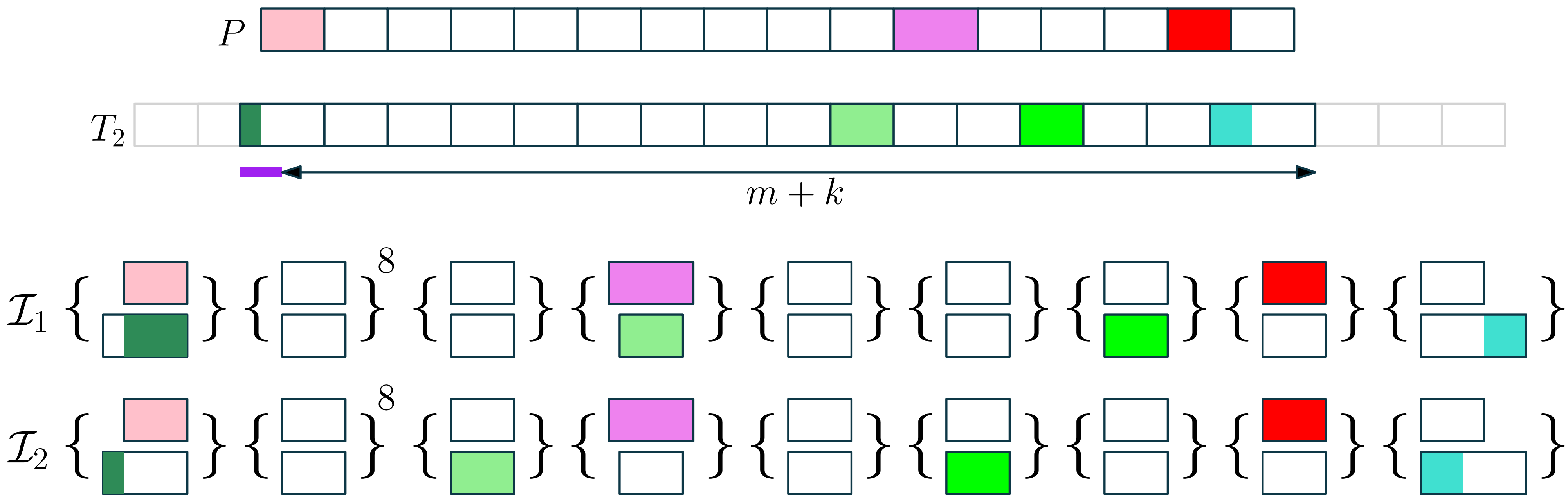


We only need to update $\mathcal{O}(k)$ pairs; there has to be a pair $\neq (Q, Q)$ involved!

Over the $\Theta(\sqrt{m})$ shifts of P, we need $\mathcal{O}(\sqrt{m} \cdot k)$ DPM-updates.

Using Dynamic Puzzle Matching

Think of: $k = 4$ and $|Q| \approx \sqrt{m}$.



We only need to update $\mathcal{O}(k)$ pairs; there has to be a pair $\neq (Q, Q)$ involved!

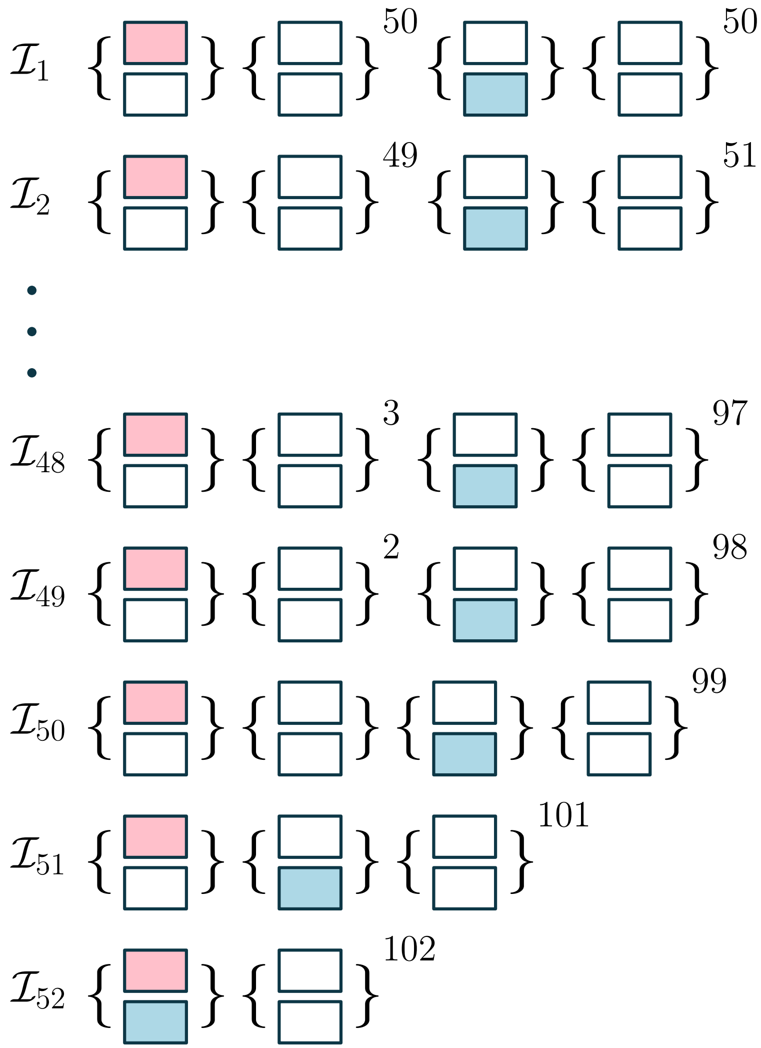
Over the $\Theta(\sqrt{m})$ shifts of P, we need $\mathcal{O}(\sqrt{m} \cdot k)$ DPM-updates.

Yields $\tilde{\mathcal{O}}(k^3 + \sqrt{m} \cdot k^2)$.

$\mathcal{O}(k^3)$ DPM-updates via Primitivity

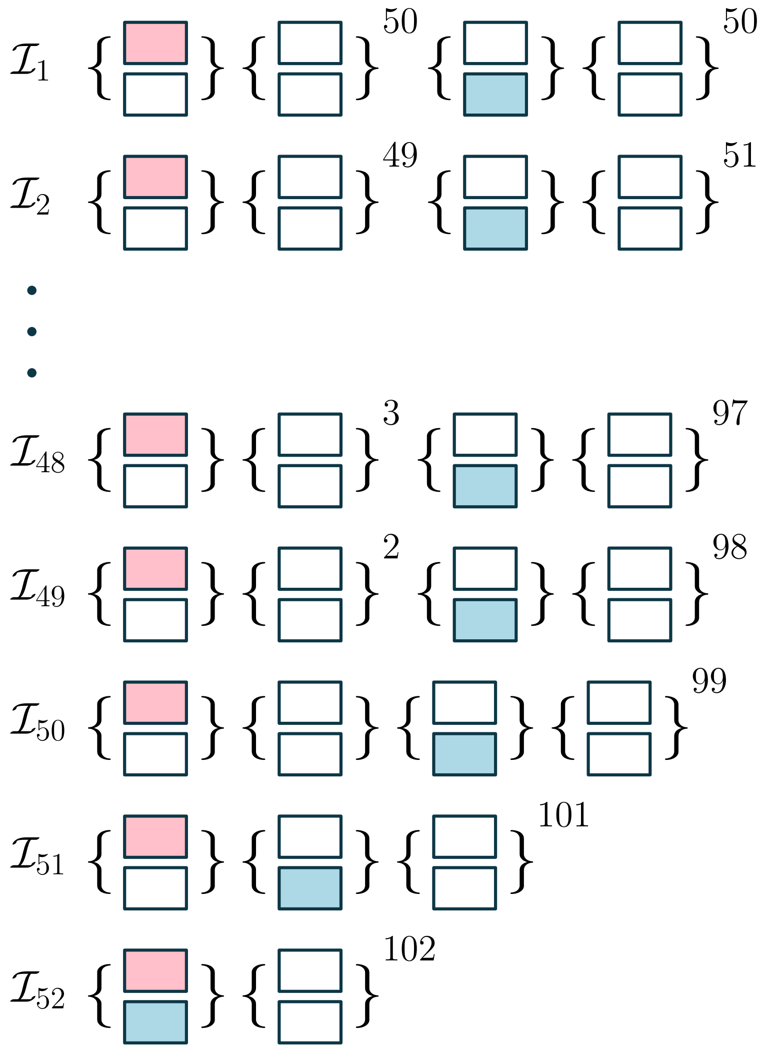
$\mathcal{O}(k^3)$ DPM-updates via Primitivity

$k = 2$



$\mathcal{O}(k^3)$ DPM-updates via Primitivity

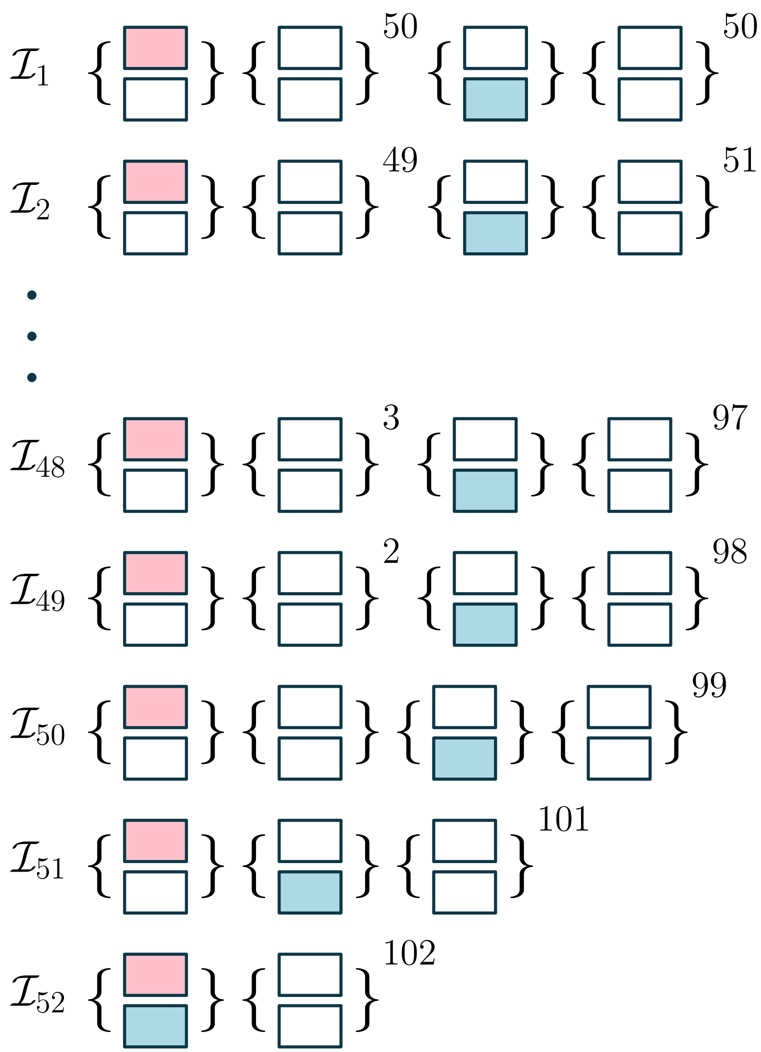
$k = 2$



For a plain run $(Q, Q)^y$, at least $y - k$ copies of Q will be matched exactly in a k-error occurrence.

$\mathcal{O}(k^3)$ DPM-updates via Primitivity

$k = 2$

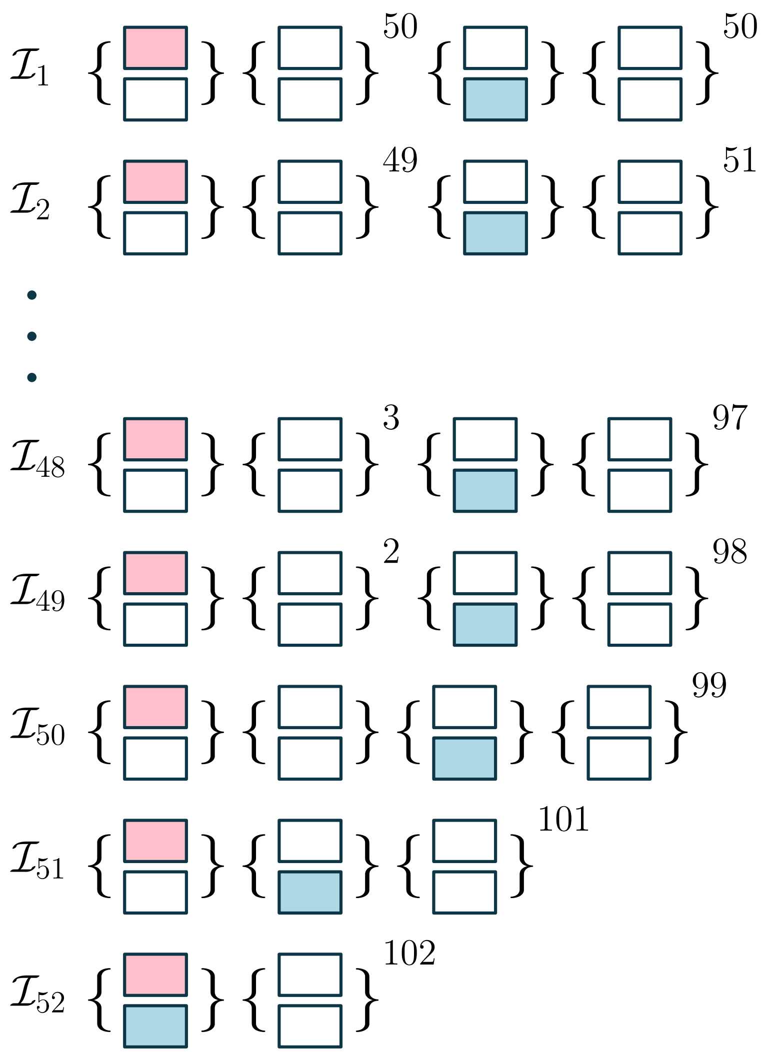


For a plain run $(Q, Q)^y$, at least $y - k$ copies of Q will be matched exactly in a k-error occurrence.

Cap exponents of plain runs at $k + 1$.

$\mathcal{O}(k^3)$ DPM-updates via Primitivity

$k = 2$



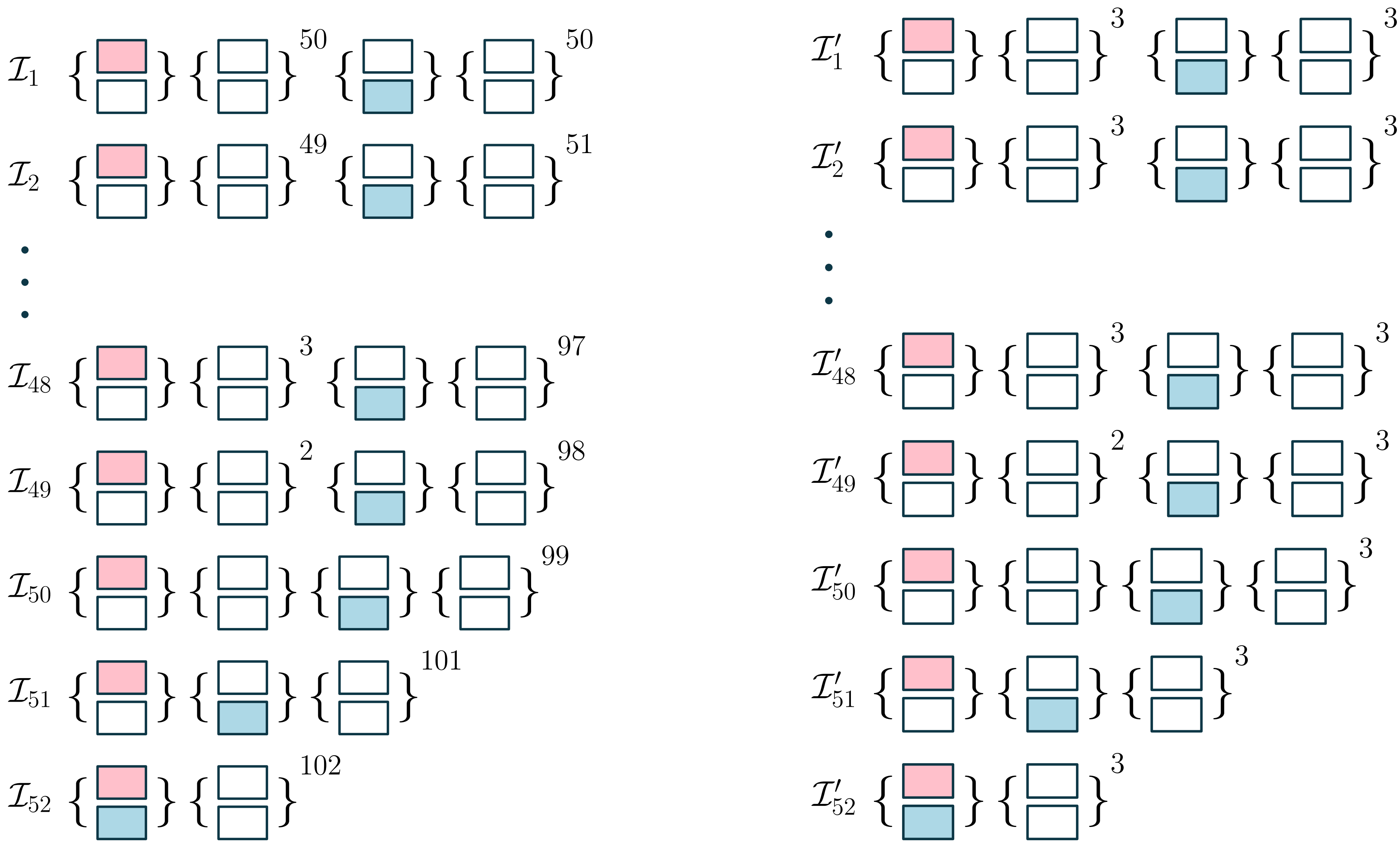
For a plain run $(Q, Q)^y$, at least $y - k$ copies of Q will be matched exactly in a k-error occurrence.

Cap exponents of plain runs at $k + 1$.

We do not lose or gain any k-error occs.

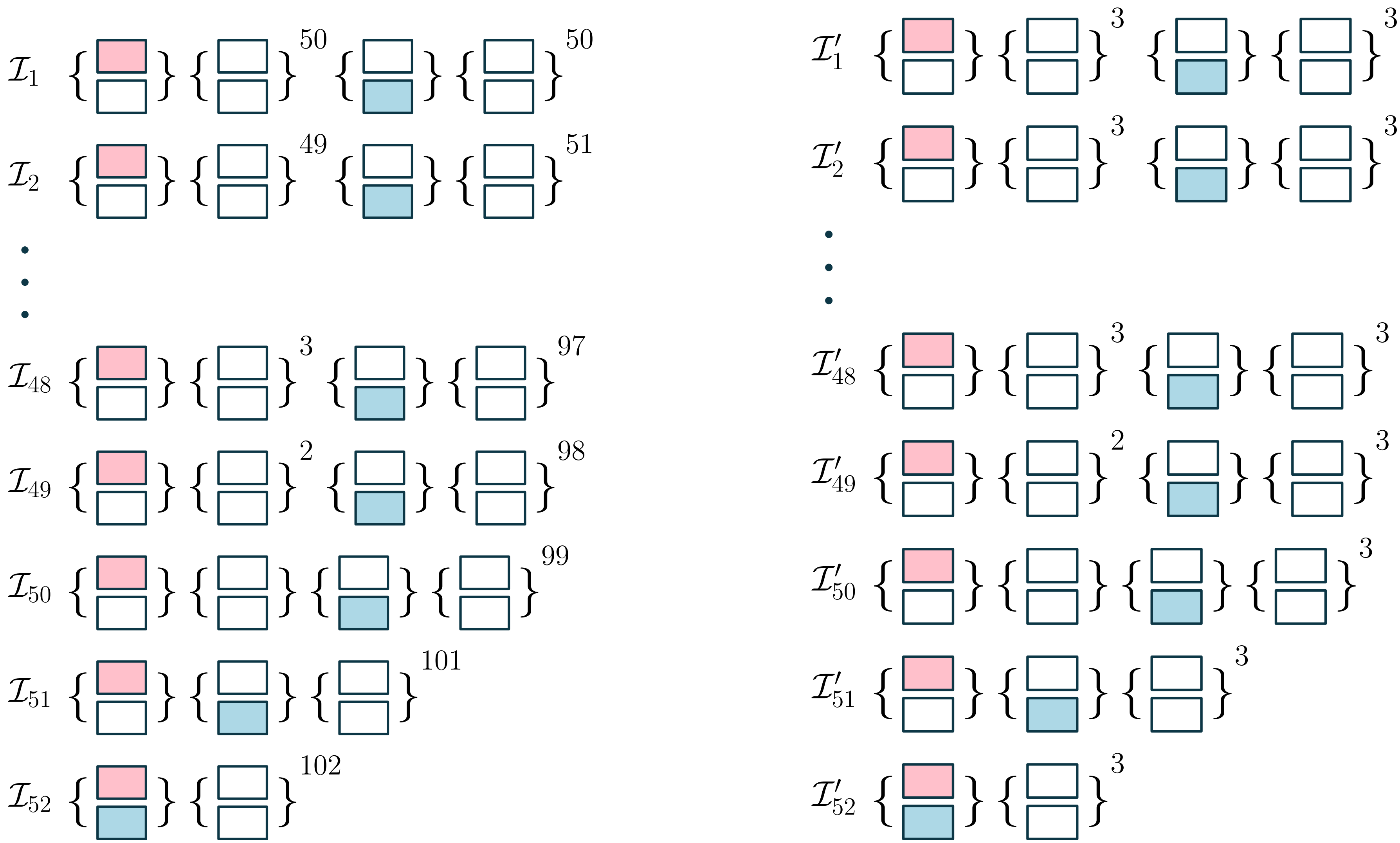
$\mathcal{O}(k^3)$ DPM-updates via Primitivity

$k = 2$



$\mathcal{O}(k^3)$ DPM-updates via Primitivity

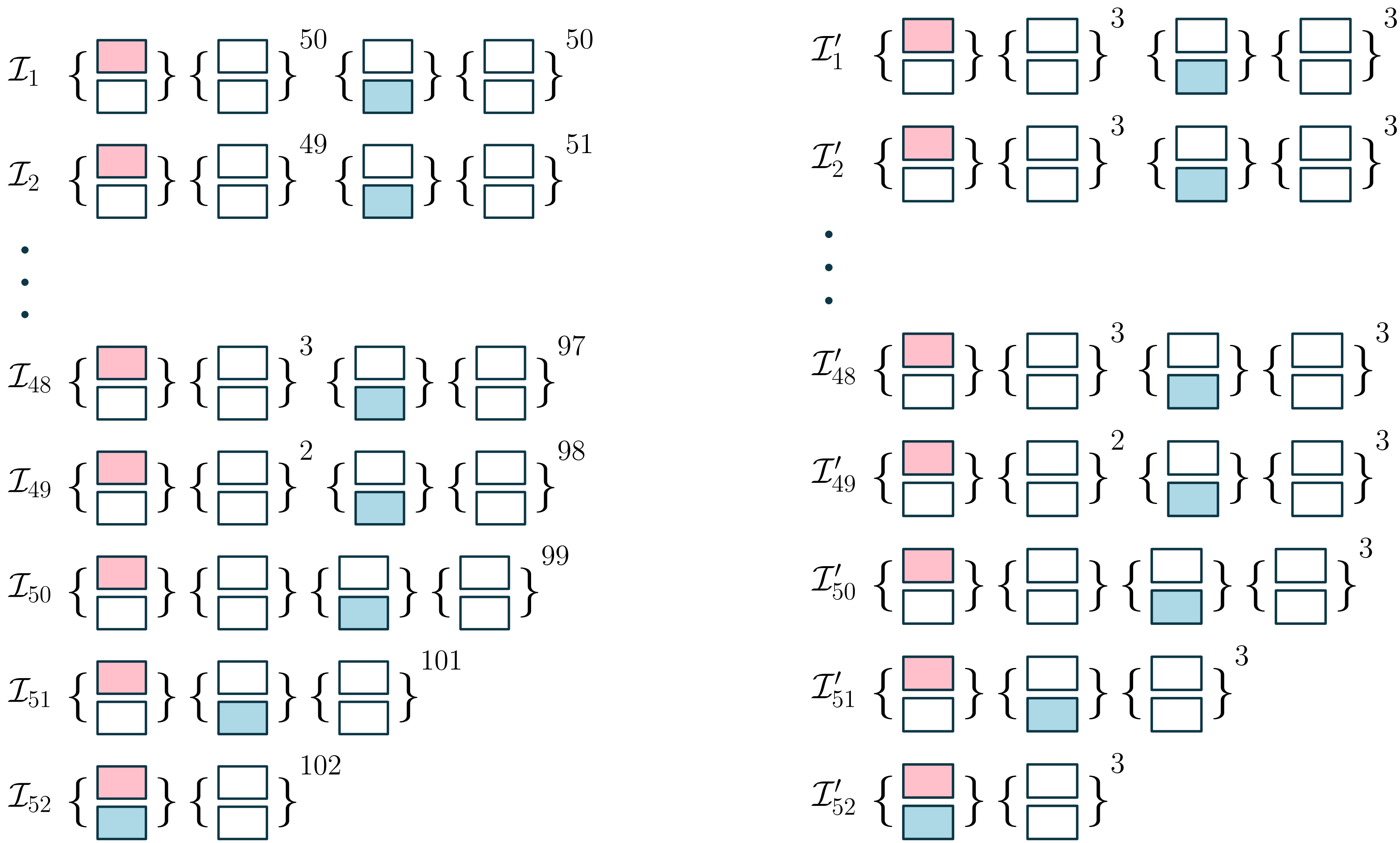
$k = 2$



The shown pair of special tiles implies $\mathcal{O}(k)$ DPM-updates.

$\mathcal{O}(k^3)$ DPM-updates via Primitivity

$k = 2$

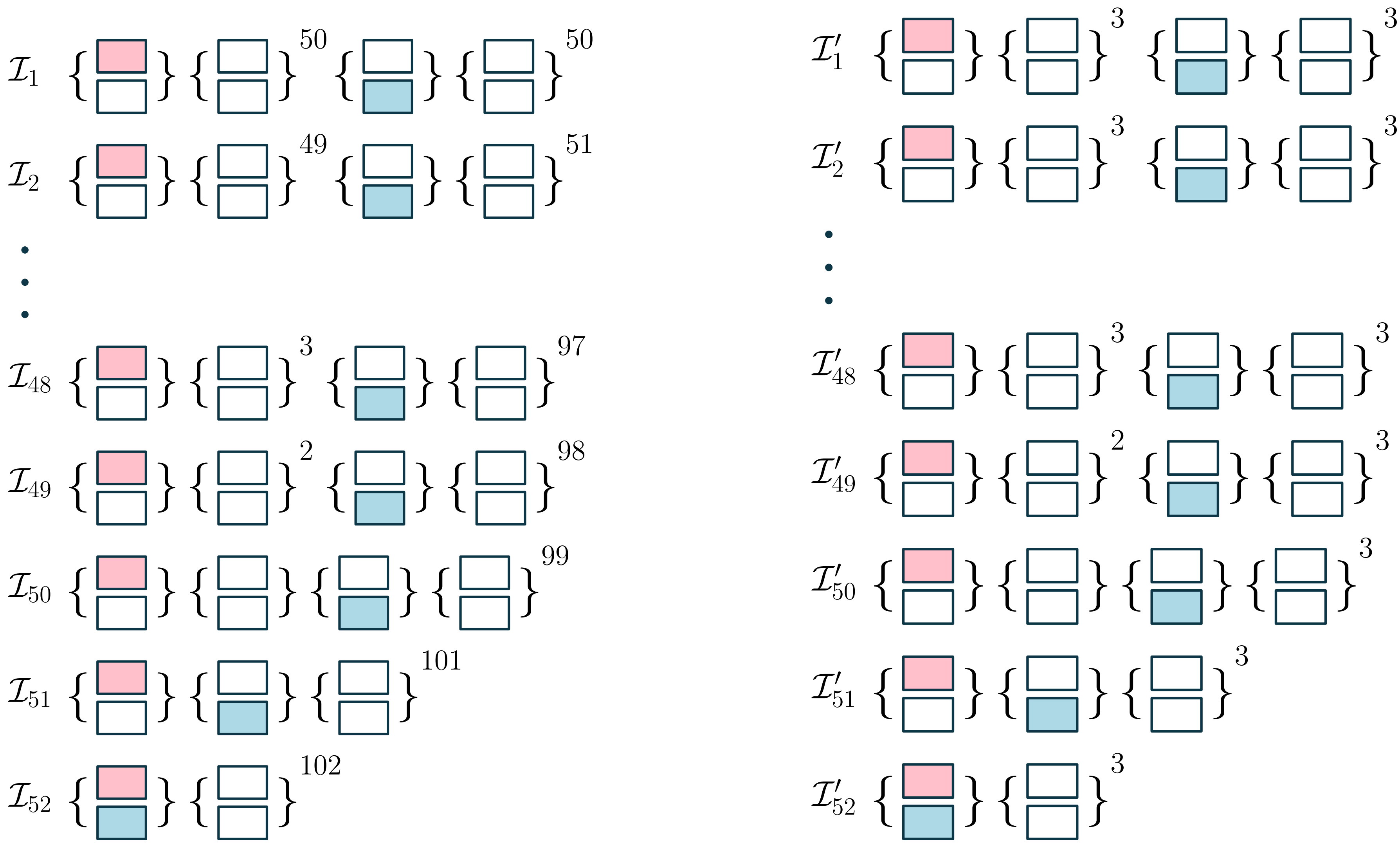


The shown pair of special tiles implies $\mathcal{O}(k)$ DPM-updates.

We have $\mathcal{O}(k^2)$ pairs of special tiles!

$\mathcal{O}(k^3)$ DPM-updates via Primitivity

$k = 2$



Alternative $\tilde{\mathcal{O}}(k^4)$-time algorithm!

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

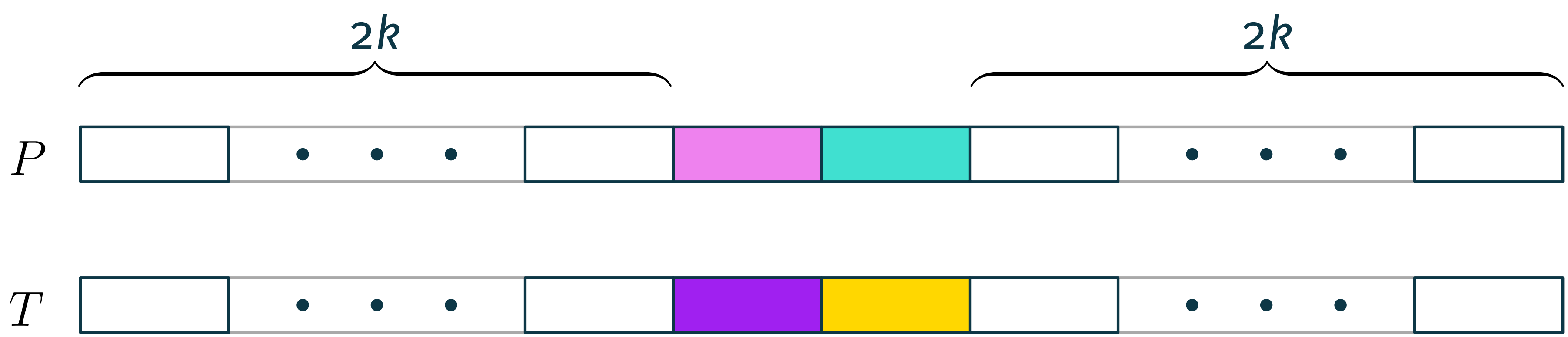
Cap exponents of plain runs at $\sqrt{k}$.

We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

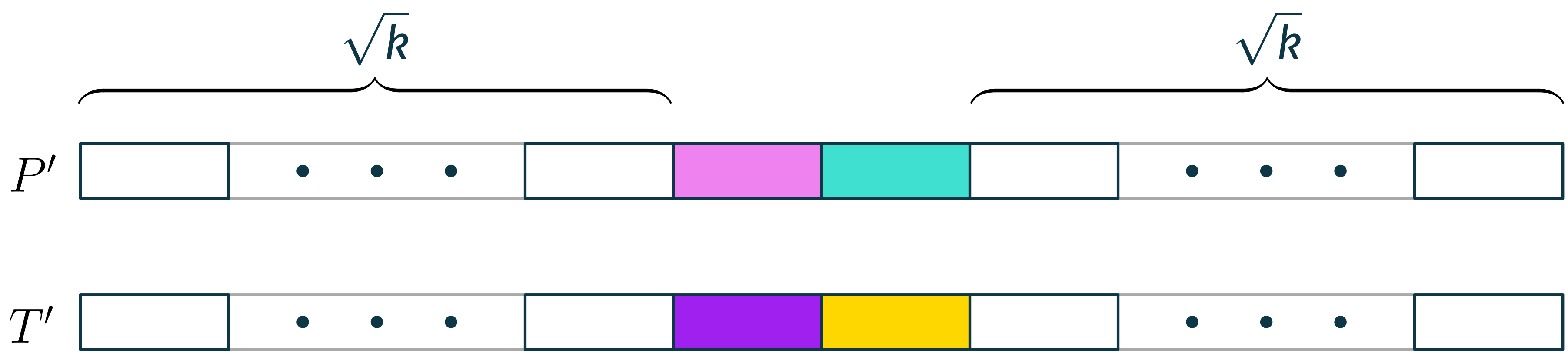
We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).



Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

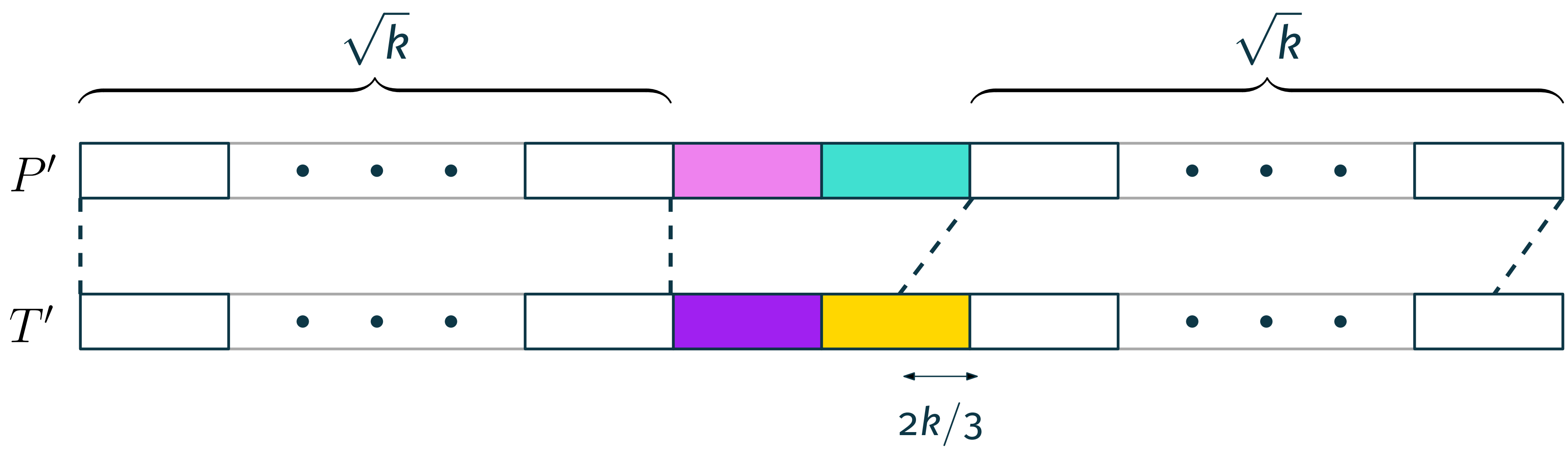
We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).



Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).



Cost: $0 + (k/3 + 1) + \sqrt{k} \cdot \delta_E(Q, rot^{2k/3}(Q))$.

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).

In this case, we must be saving $\geq \sqrt{k}$ by canceling out errors between P and Q^∞ with errors between T and Q^∞.

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).

In this case, we must be saving $\geq \sqrt{k}$ by canceling out errors between P and Q^∞ with errors between T and Q^∞.

We quantify potential savings using a marking scheme based on overlaps of special tiles and verify $\mathcal{O}(k^{2.5})$ positions with $\geq \sqrt{k}$ marks using known techniques.

Overview for $\mathcal{O}(k^{2.5})$ DPM-updates

Cap exponents of plain runs at $\sqrt{k}$.

We may get false positives when we have $\geq \sqrt{k}$ edits in a run of (Q, Q).

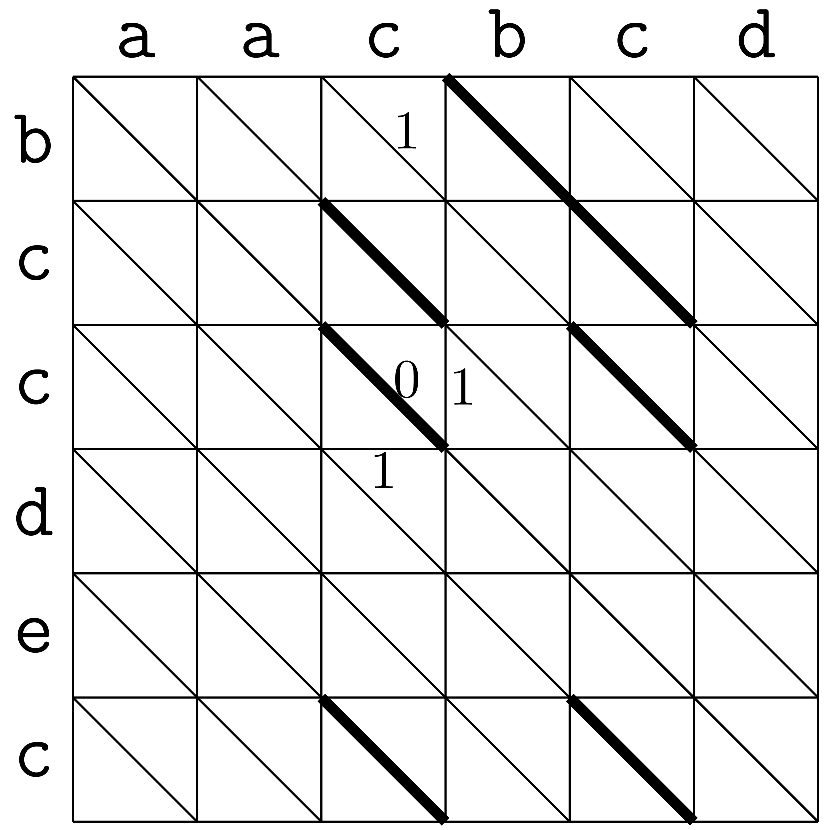
In this case, we must be saving $\geq \sqrt{k}$ by canceling out errors between P and Q^∞ with errors between T and Q^∞.

We quantify potential savings using a marking scheme based on overlaps of special tiles and verify $\mathcal{O}(k^{2.5})$ positions with $\geq \sqrt{k}$ marks using known techniques.

This yields $\mathcal{O}(k^{2.5})$ DPM-updates and hence $\tilde{\mathcal{O}}(k^{3.5})$ time overall.

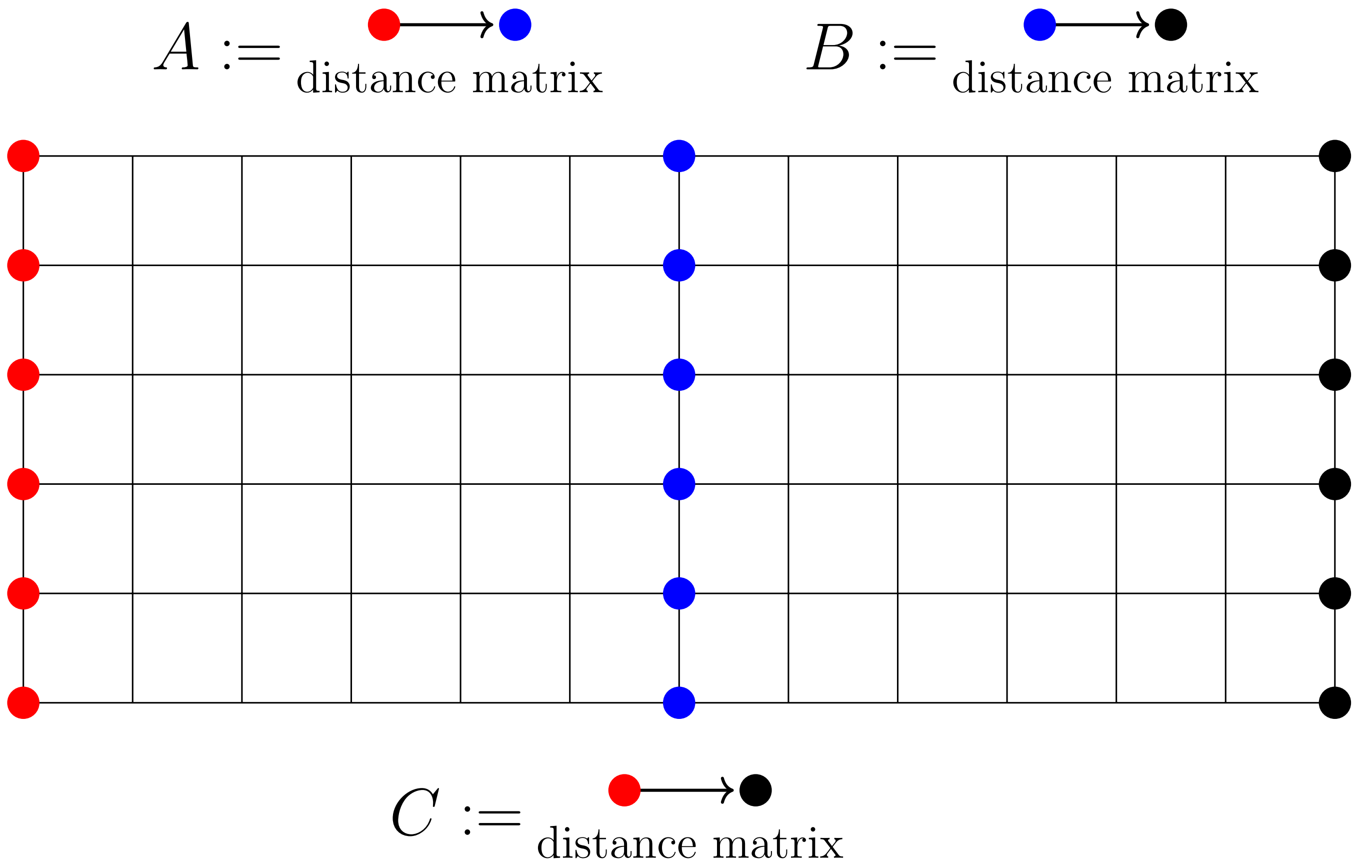
A Solution to DPM and a Grid View

A Solution to DPM and a Grid View

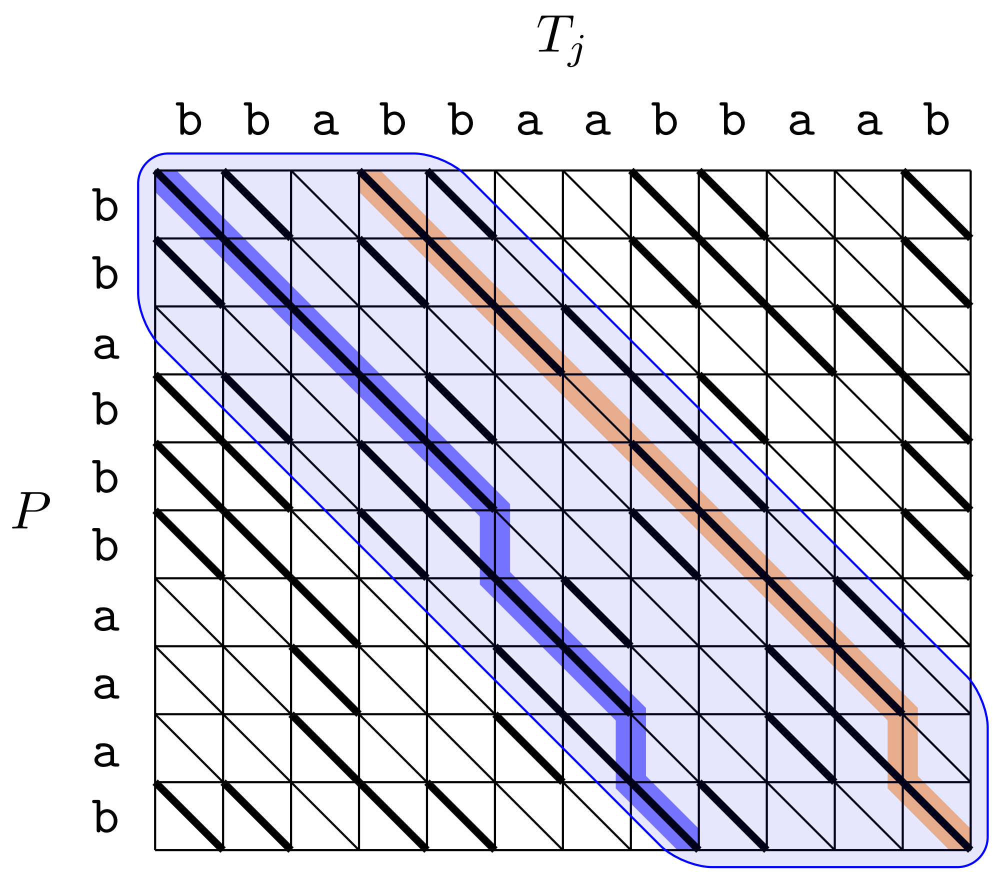


A Solution to DPM and a Grid View

Theorem [Tiskin; Algorithmica 2015] Matrix C can be computed from (small representations of) $n \times n$ matrices A and B in $\mathcal{O}(n \log n)$ time.



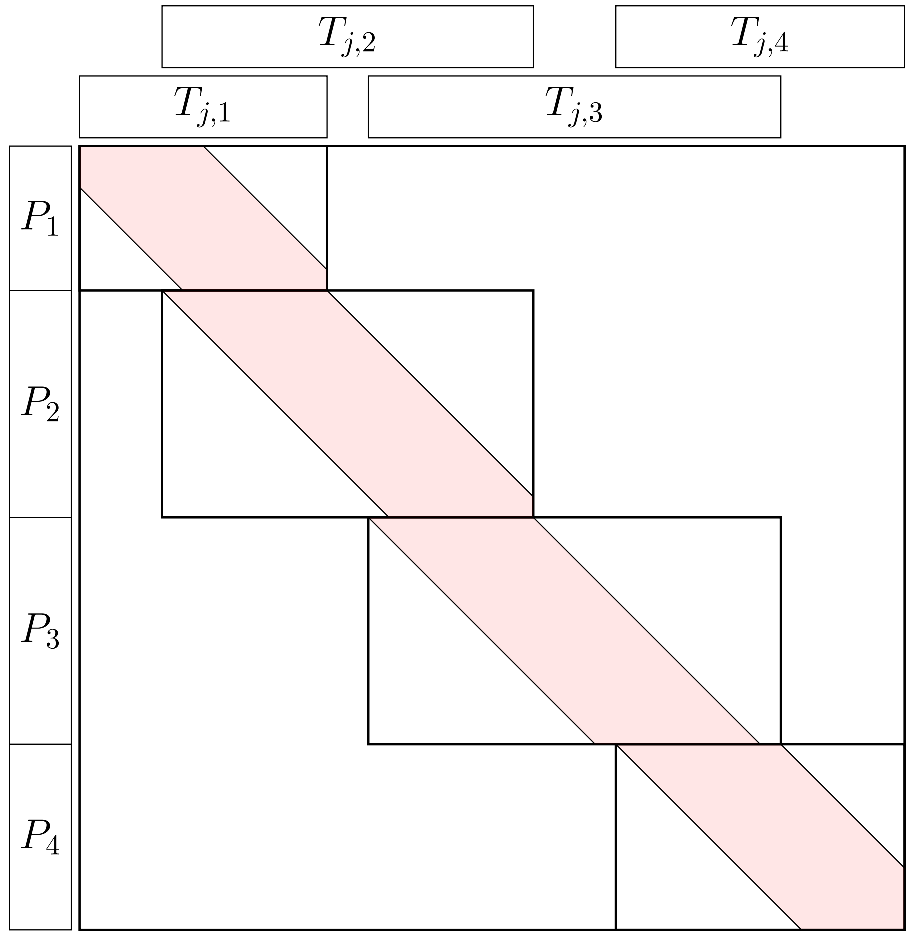
A Solution to DPM and a Grid View



$$P = 10,\ T_j = 12,\ k = 2.$$

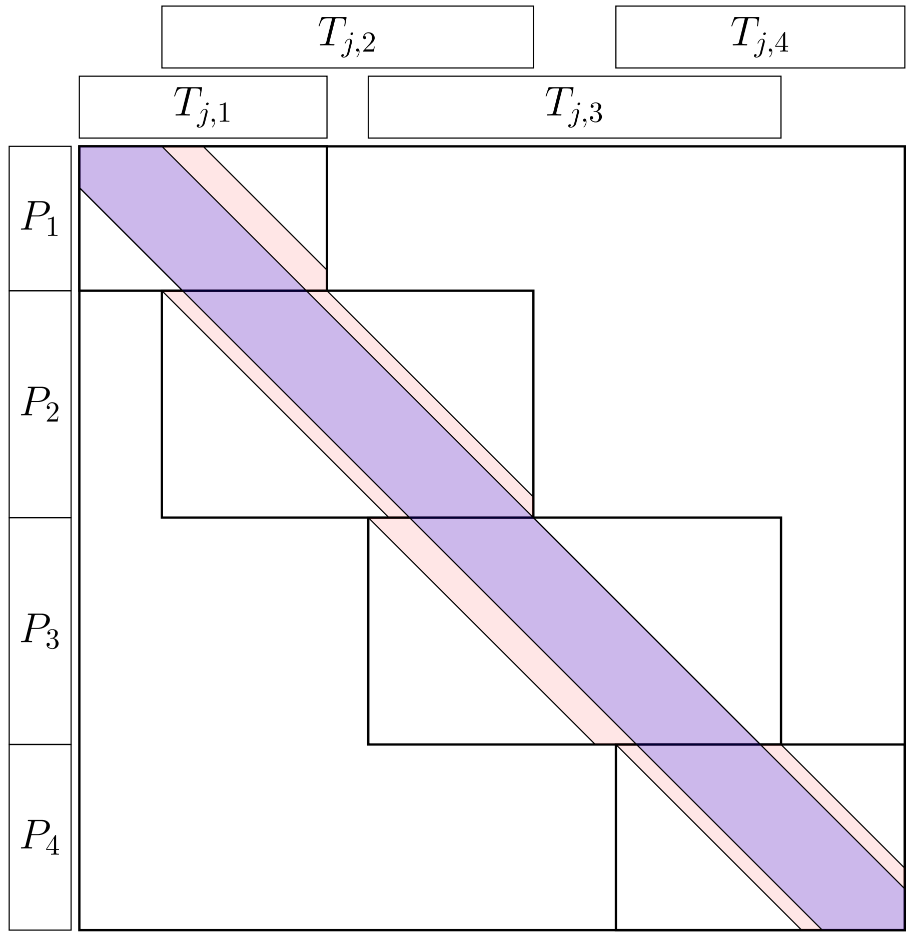
Only $|T_j| - |P| + 2k + 1 = \mathcal{O}(k)$ diagonals are relevant.

A Solution to DPM and a Grid View



Preprocessing: Build distance matrices for these small alignment grids.

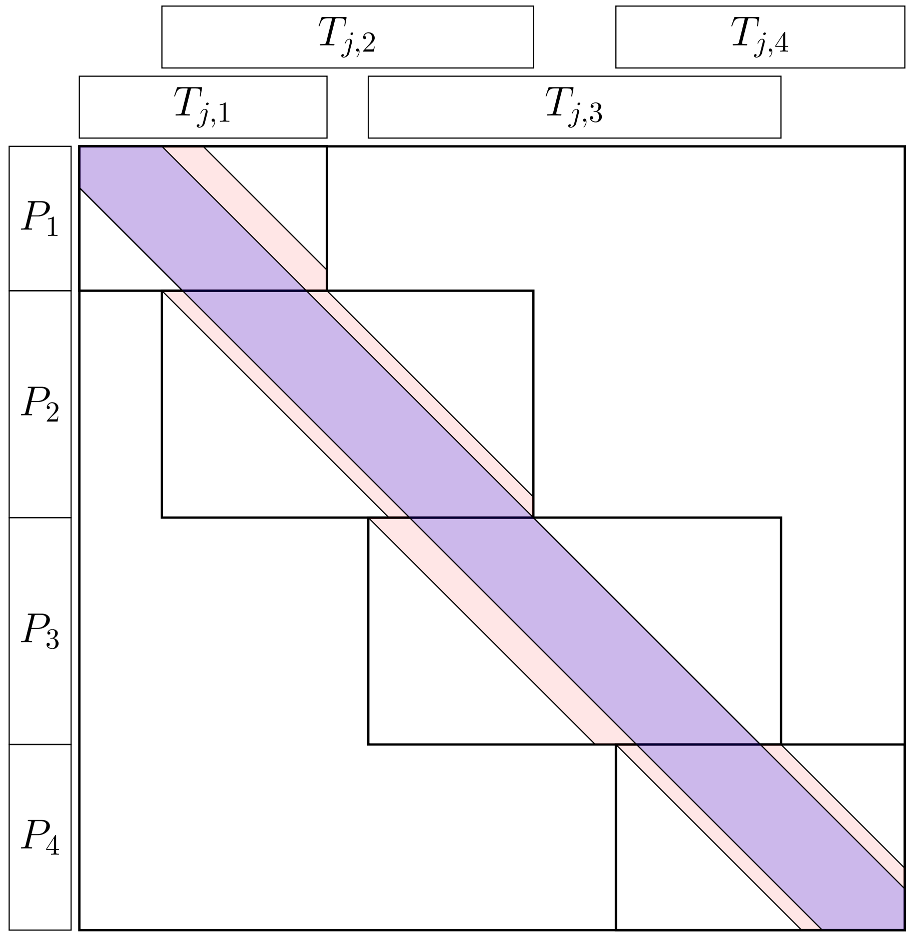
A Solution to DPM and a Grid View



Preprocessing: Build distance matrices for these small alignment grids.

Update: Maintain a balanced binary tree over them, stitching them together.

A Solution to DPM and a Grid View



Preprocessing: Build distance matrices for these small alignment grids.

Update: Maintain a balanced binary tree over them, stitching them together.

Each stitching operation takes $\tilde{O}(k)$ time.

Final Remarks and Open Problems

Final Remarks and Open Problems

What is the right exponent?

Cole and Hariharan's conjecture: $\mathcal{O}(n + k^3 \cdot n/m)$ *should be possible.*

Final Remarks and Open Problems

What is the right exponent?

Cole and Hariharan's conjecture: $\mathcal{O}(n + k^3 \cdot n/m)$ *should be possible.*

Is the decision version easier?

Final Remarks and Open Problems

What is the right exponent?

Cole and Hariharan's conjecture: $\mathcal{O}(n + k^3 \cdot n/m)$ *should be possible.*

Is the decision version easier?

What if we allow for some approximation by also reporting an arbitrary subset of the positions in $\mathrm{Occ}^{E}_{(1+\epsilon)k}(P, T) \setminus \mathrm{Occ}^{E}_{k}(P, T)$ for a small $\epsilon > 0$?

Final Remarks and Open Problems

What is the right exponent?

Cole and Hariharan's conjecture: $\mathcal{O}(n + k^3 \cdot n/m)$ *should be possible.*

Is the decision version easier?

What if we allow for some approximation by also reporting an arbitrary subset of the positions in $\mathrm{Occ}^E_{(1+\epsilon)k}(P, T) \setminus \mathrm{Occ}^E_k(P, T)$ for a small $\epsilon > 0$?

We report starting positions. How fast can we report substrings?

The End

Thank you for your attention!

Many thanks to Philip Wellnitz for sharing his slides from SODA 2019!

I have edited the portion I used, so I am responsible for any errors. :)