Karp-Rabin and Streaming Pattern Matching

Jakub Radoszewski

University of Warsaw, Poland

Text Algorithms

Pattern Matching

Problem (Pattern Matching)

Given two strings, a text $t[0..n-1]$ and a pattern $p[0..m-1]$, find all occurrences of the pattern p in the text t.

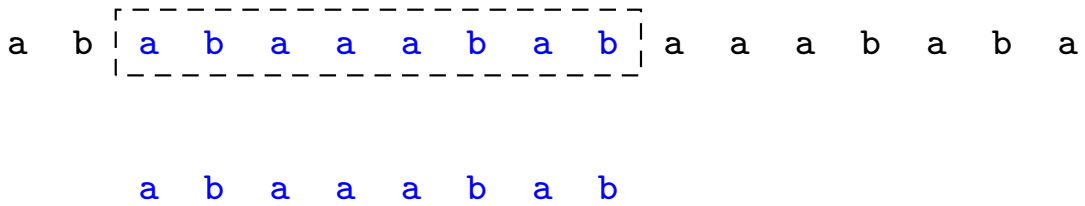
a b a b a a a b a b a a a b a b a

a b a a a b a b

Pattern Matching

Problem (Pattern Matching)

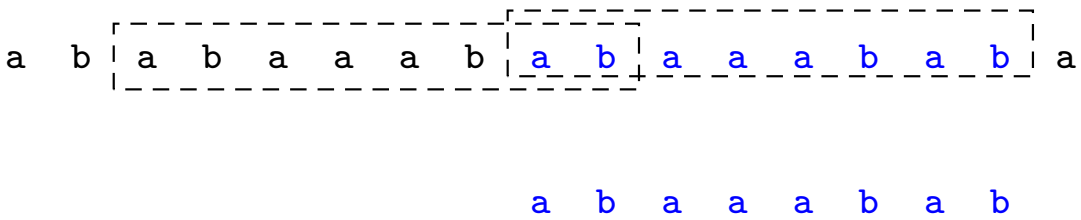
Given two strings, a text $t[0..n-1]$ and a pattern $p[0..m-1]$, find all occurrences of the pattern p in the text t.



Pattern Matching

Problem (Pattern Matching)

Given two strings, a text $t[0..n-1]$ and a pattern $p[0..m-1]$, find all occurrences of the pattern p in the text t.



String Hashing

For a string s of length m, we use the **Karp-Rabin** hash function (*polynomial hash*, *rolling hash*):

$$h(s) = (s[0]r^{m-1} + s[1]r^{m-2} + \ldots + s[m-2]r + s[m-1]) \bmod q.$$

Here r and q are integer parameters.

String Hashing

For a string s of length m, we use the **Karp-Rabin** hash function (*polynomial hash*, *rolling hash*):

$$h(s) = (s[0]r^{m-1} + s[1]r^{m-2} + \ldots + s[m-2]r + s[m-1]) \bmod q.$$

Here r and q are integer parameters.

Theorem

If q is a prime number, the probability that for a random r two strings of the same length m receive the same hash (i.e., $h(s) = h(t)$ for $s \neq t$) is $\leqslant \frac{m}{q-1}$.

String Hashing

For a string s of length m, we use the **Karp-Rabin** hash function (*polynomial hash*, *rolling hash*):

$$h(s) = (s[0]r^{m-1} + s[1]r^{m-2} + \ldots + s[m-2]r + s[m-1]) \bmod q.$$

Here r and q are integer parameters.

Theorem

If q is a prime number, the probability that for a random r two strings of the same length m receive the same hash (i.e., $h(s) = h(t)$ for $s \neq t$) is $\leqslant \frac{m}{q-1}$.

Conclusion: One needs to properly choose q and r.

Computing String Hashes

Denote:

$$H_i = h(s[0..i]).$$

Then:

$$\begin{aligned} H_0 &= s[0] \bmod q \\ H_1 &= (s[0]r + s[1]) \bmod q \\ H_2 &= (s[0]r^2 + s[1]r + s[2]) \bmod q \\ &\dots \end{aligned}$$

Computing String Hashes

Denote:

$$H_i = h(s[0..i]).$$

Then:

$$\begin{aligned} H_0 &= s[0] \bmod q \\ H_1 &= (s[0]r + s[1]) \bmod q \\ H_2 &= (s[0]r^2 + s[1]r + s[2]) \bmod q \\ &\cdots \\ H_{i+1} &= (H_i r + s[i+1]) \bmod q \end{aligned}$$

Computing String Hashes

Denote:

$$H_i = h(s[0..i]).$$

Then:

$$\begin{aligned} H_0 &= s[0] \bmod q \\ H_1 &= (s[0]r + s[1]) \bmod q \\ H_2 &= (s[0]r^2 + s[1]r + s[2]) \bmod q \\ &\cdots \\ H_{i+1} &= (H_i r + s[i+1]) \bmod q \end{aligned}$$

Thus we can compute $H_{m-1} = h(s)$ in $O(m)$ time and $O(1)$ space.

Rolling Property

Denote:

$$H_{i,j} = h(s[i..j]).$$

Then:

$$\begin{aligned} H_{i,j} &= (s[i]r^{j-i} + s[i+1]r^{j-i-1} \ldots + s[j]) \bmod q \\ H_{i+1,j+1} &= (s[i+1]r^{j-i} + \ldots + s[j]r + s[j+1]) \bmod q \end{aligned}$$

$$H_{i+1,j+1} = ((H_{i,j} - s[i]r^{j-i}) \cdot r + s[j+1]) \bmod q$$

Rolling Property

Denote:

$$H_{i,j} = h(s[i..j]).$$

Then:

$$\begin{aligned} H_{i,j} &= (s[i]r^{j-i} + s[i+1]r^{j-i-1} \ldots + s[j]) \bmod q \\ H_{i+1,j+1} &= (s[i+1]r^{j-i} + \ldots + s[j]r + s[j+1]) \bmod q \end{aligned}$$

$$H_{i+1,j+1} = ((H_{i,j} - s[i]r^{j-i}) \cdot r + s[j+1]) \bmod q$$

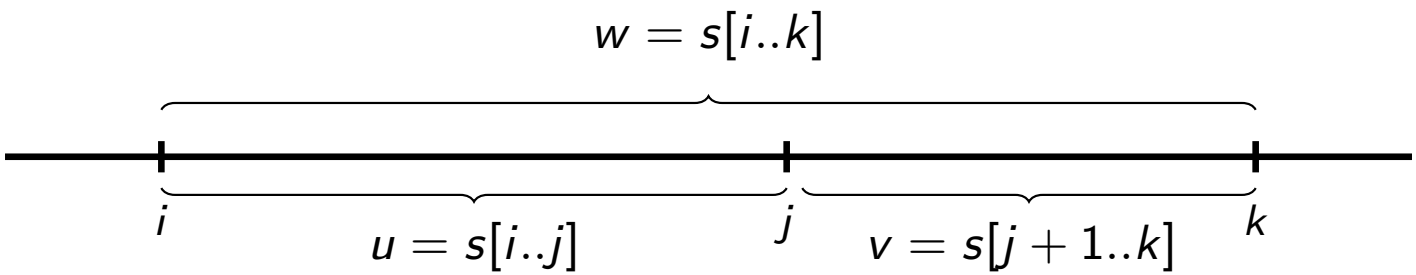
The power r^a can be computed in $O(\log a)$ time using doubling.

Karp-Rabin Pattern Matching

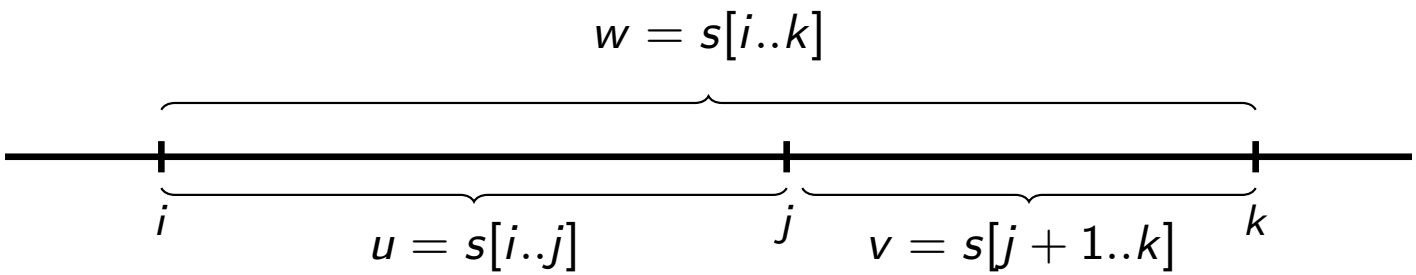
Idea of the algorithm:

1. Compute the hash of the pattern, $h(p)$
2. Compute the rolling hashes of all substrings of the text of length m, $h(t[i-m+1..i]) = H_{i-m+1,i}$
3. If any of the hashes is equal to $h(p)$:
   (a) Either report an occurrence at position i (may incur false matches with a small probability)
   (b) Or check if this is indeed an occurrence assiduously (has $O(nm)$-time behavior in the worst case)

A Property of String Hashes

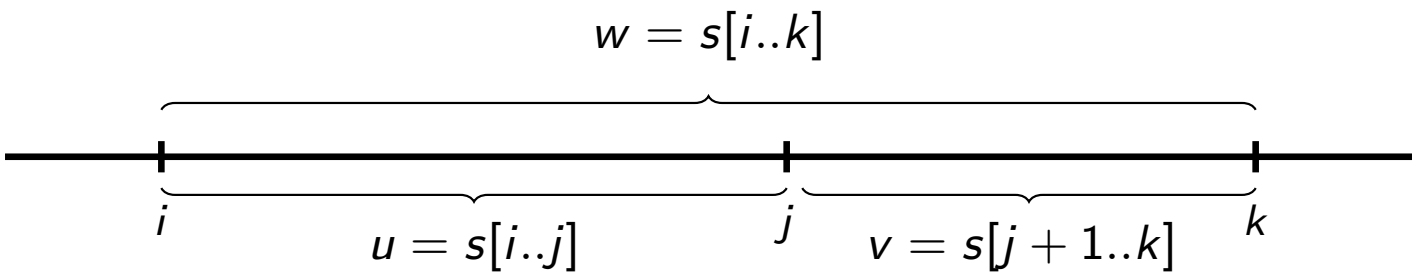


A Property of String Hashes



$$h(w) = (h(u) \cdot r^{k-j} + h(v)) \bmod q$$

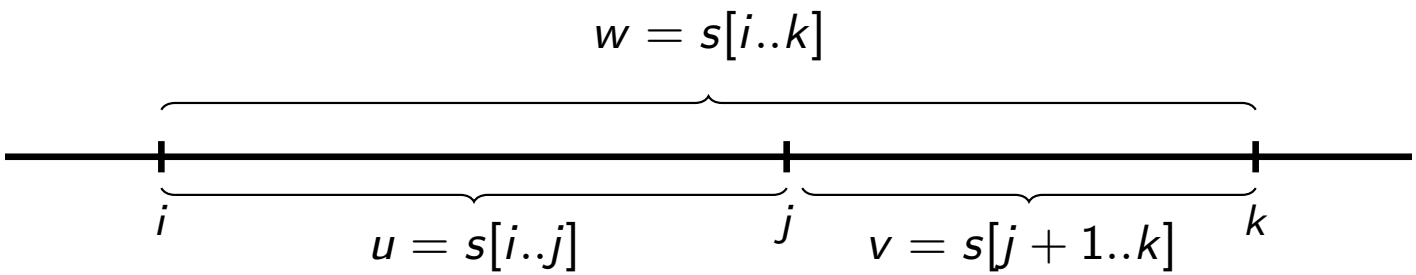
A Property of String Hashes



$$h(w) = (h(u) \cdot r^{k-j} + h(v)) \bmod q$$

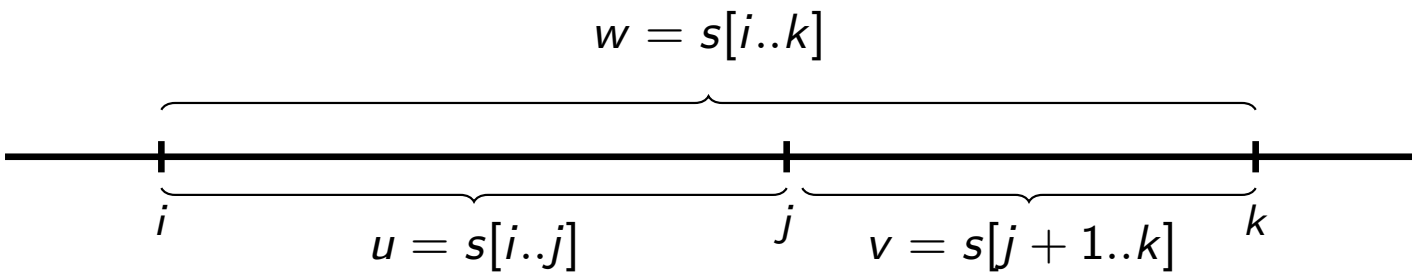
$$h(v) = (h(w) - h(u) \cdot r^{k-j}) \bmod q$$

A Property of String Hashes



$$h(w) = (h(u) \cdot r^{k-j} + h(v)) \bmod q$$
$$h(v) = (h(w) - h(u) \cdot r^{k-j}) \bmod q$$
$$h(u) = ((h(w) - h(v)) \cdot (r^{-1})^{k-j}) \bmod q$$

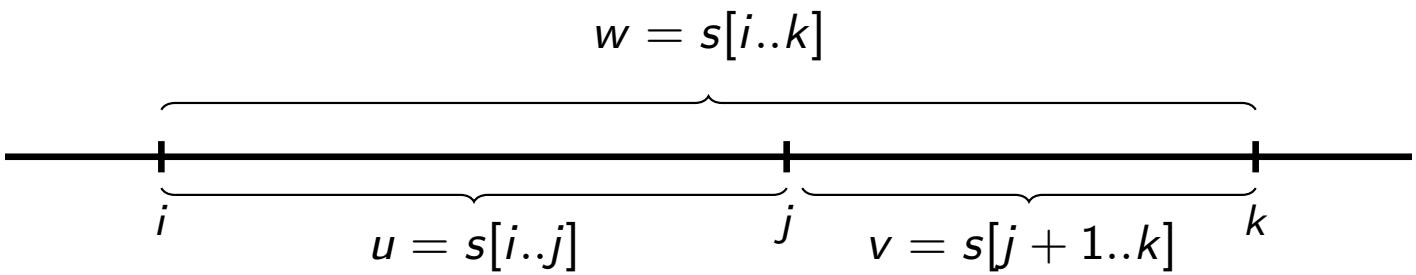
A Property of String Hashes



$$h(w) = (h(u) \cdot r^{k-j} + h(v)) \bmod q$$
$$h(v) = (h(w) - h(u) \cdot r^{k-j}) \bmod q$$
$$h(u) = ((h(w) - h(v)) \cdot (r^{-1})^{k-j}) \bmod q$$

$r^{-1} \bmod q$ can be precomputed in $O(\log q)$ time.

A Property of String Hashes



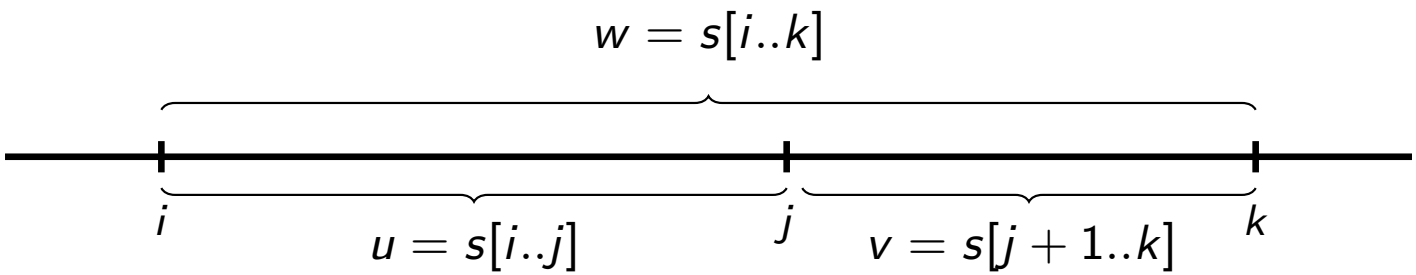
$$h(w) = (h(u) \cdot r^{k-j} + h(v)) \bmod q$$
$$h(v) = (h(w) - h(u) \cdot r^{k-j}) \bmod q$$
$$h(u) = ((h(w) - h(v)) \cdot (r^{-1})^{k-j}) \bmod q$$

$r^{-1} \bmod q$ can be precomputed in $O(\log q)$ time.

Lemma

Knowing the hashes of any two of the three strings u, v, w, we can compute the hash of the third in constant time.

A Property of String Hashes



$$h(w) = (h(u) \cdot r^{k-j} + h(v)) \bmod q$$
$$h(v) = (h(w) - h(u) \cdot r^{k-j}) \bmod q$$
$$h(u) = ((h(w) - h(v)) \cdot (r^{-1})^{k-j}) \bmod q$$

$r^{-1} \bmod q$ can be precomputed in $O(\log q)$ time.

Lemma

Knowing the hashes of any two of the three strings u, v, w, we can compute the hash of the third in constant time.

Trick: For a string x of length m, together with its hash, we store $r^m \bmod q$ and $(r^{-1})^m \bmod q$.

Pattern Matching in the Streaming Model

Streaming model:

- It was introduced for the purposes of processing massive data in small space
- We cannot afford to store the whole input data
- The input data arrives as a stream of items. It is read only once

Example 1: Missing Number

Problem.
A sequence of numbers is given; it contains all numbers from $\{1, \ldots, n\}$ (in some order) except for one number. Find the missing number!

Example. $n = 7$
6, 3, 4, 1, 7, 2

Example 1: Missing Number

Problem.
A sequence of numbers is given; it contains all numbers from $\{1, \ldots, n\}$ (in some order) except for one number. Find the missing number!

Example. $n = 7$
6, 3, 4, 1, 7, 2 $\longrightarrow$ **5** is missing!

Streaming Solution

Goal: Use $O(1)$ space.

Streaming Solution

Goal: Use $O(1)$ space.

a — Missing number
S — Sum of input numbers

Streaming Solution

Goal: Use $O(1)$ space.

a — Missing number
S — Sum of input numbers

$S + a = 1 + \ldots + n$

Streaming Solution

Goal: Use $O(1)$ space.

a — Missing number
S — Sum of input numbers

$S + a = 1 + \ldots + n$
So $a = \frac{n(n+1)}{2} - S$

Example 2: Two Missing Numbers

Problem.
A sequence of numbers is given; it contains all numbers from $\{1, \ldots, n\}$ (in some order) except for two numbers. Find the two missing numbers!

Example. $n = 7$
6, 4, 1, 7, 2

Example 2: Two Missing Numbers

Problem.
A sequence of numbers is given; it contains all numbers from $\{1, \ldots, n\}$ (in some order) except for two numbers. Find the two missing numbers!

Example. $n = 7$
6, 4, 1, 7, 2 $\longrightarrow$ **3** and **5** are missing

Solution

a, b — missing numbers
S_1 — sum of input numbers

Solution

a, b — missing numbers
S_1 — sum of input numbers

$$S_1 + a + b = 1 + \ldots + n = \frac{n(n+1)}{2}$$

Solution

a, b — missing numbers
S_1 — sum of input numbers
S_2 — sum of squares of input numbers

$$S_1 + a + b = 1 + \ldots + n = \frac{n(n+1)}{2}$$

Solution

a, b — missing numbers
S_1 — sum of input numbers
S_2 — sum of squares of input numbers

$$S_1 + a + b = 1 + \ldots + n = \frac{n(n+1)}{2}$$
$$S_2 + a^2 + b^2 = 1^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Solution

a, b — missing numbers
S_1 — sum of input numbers
S_2 — sum of squares of input numbers

$$S_1 + a + b = 1 + \ldots + n = \frac{n(n+1)}{2}$$
$$S_2 + a^2 + b^2 = 1^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Then:

$$a + b = \frac{n(n+1)}{2} - S_1$$
$$a^2 + b^2 = \frac{n(n+1)(2n+1)}{6} - S_2$$

Solution

a, b — missing numbers
S_1 — sum of input numbers
S_2 — sum of squares of input numbers

$$S_1 + a + b = 1 + \ldots + n = \frac{n(n+1)}{2}$$
$$S_2 + a^2 + b^2 = 1^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Then:

$$a + b = \frac{n(n+1)}{2} - S_1 = X$$
$$a^2 + b^2 = \frac{n(n+1)(2n+1)}{6} - S_2 = Y$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases}$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ a^2 + (X - a)^2 = Y \end{cases}$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - 2aX + X^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ a^2 + (X - a)^2 = Y \end{cases}$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ a^2 + (X - a)^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - 2aX + X^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - (2X)a + (X^2 - Y) = 0 \end{cases}$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ a^2 + (X - a)^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - 2aX + X^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - (2X)a + (X^2 - Y) = 0 \end{cases}$$

$$\Delta = 4X^2 - 8(X^2 - Y) = 8Y - 4X^2$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases} \qquad \begin{cases} b = X - a \\ a^2 + (X - a)^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - 2aX + X^2 = Y \end{cases} \qquad \begin{cases} b = X - a \\ 2a^2 - (2X)a + (X^2 - Y) = 0 \end{cases}$$

$$\Delta = 4X^2 - 8(X^2 - Y) = 8Y - 4X^2$$

$$\begin{cases} a_1 = \frac{2X - \sqrt{\Delta}}{4} \\ b_1 = X - a_1 \end{cases} \qquad \begin{cases} a_2 = \frac{2X + \sqrt{\Delta}}{4} \\ b_2 = X - a_2 \end{cases}$$

System of Equations

$$\begin{cases} a + b = X \\ a^2 + b^2 = Y \end{cases} \qquad \begin{cases} b = X - a \\ a^2 + (X - a)^2 = Y \end{cases}$$

$$\begin{cases} b = X - a \\ 2a^2 - 2aX + X^2 = Y \end{cases} \qquad \begin{cases} b = X - a \\ 2a^2 - (2X)a + (X^2 - Y) = 0 \end{cases}$$

$$\Delta = 4X^2 - 8(X^2 - Y) = 8Y - 4X^2$$

$$\begin{cases} a_1 = \frac{2X - \sqrt{\Delta}}{4} \\ b_1 = X - a_1 \end{cases} \qquad \begin{cases} a_2 = \frac{2X + \sqrt{\Delta}}{4} \\ b_2 = X - a_2 \end{cases}$$

Why two solutions?

Pattern Matching in the Streaming Model

- First the pattern arrives, symbol by symbol. Then the text arrives, symbol by symbol
- For convenience we assume that we know the lengths of the pattern (m) and of the text (n)
- We are aiming at total space $O(\log m)$
- The time of processing a letter from the pattern/text should be $O(\log m)$

Pattern Matching in the Streaming Model

- First the pattern arrives, symbol by symbol. Then the text arrives, symbol by symbol
- For convenience we assume that we know the lengths of the pattern (m) and of the text (n)
- We are aiming at total space $O(\log m)$
- The time of processing a letter from the pattern/text should be $O(\log m)$

Is this possible at all?

Pattern Matching in the Streaming Model

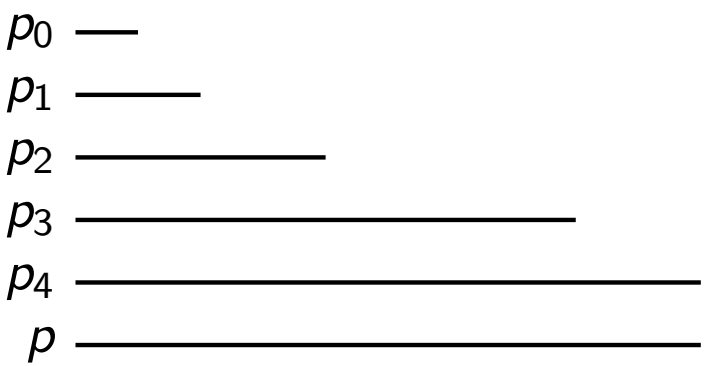
- First the pattern arrives, symbol by symbol. Then the text arrives, symbol by symbol
- For convenience we assume that we know the lengths of the pattern (m) and of the text (n)
- We are aiming at total space $O(\log m)$
- The time of processing a letter from the pattern/text should be $O(\log m)$

Is this possible at all?

No, unless randomization is allowed!

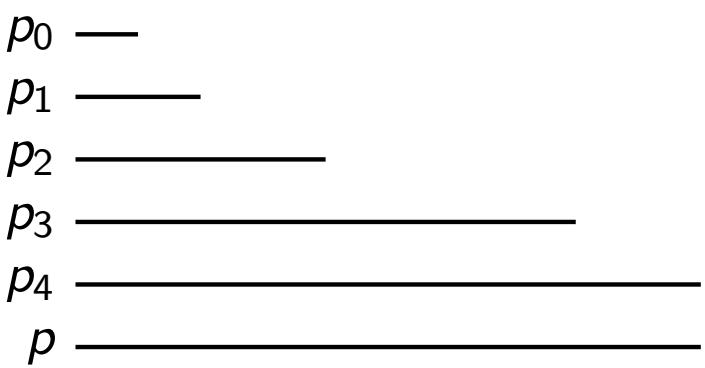
Streaming Pattern Matching, Main Idea

1. Let p_j be the prefix of the pattern of length 2^j.
   (We assume $p_k = p$ for $k = \lceil \log m \rceil$)
   Compute $h(p_i)$ when reading the pattern.

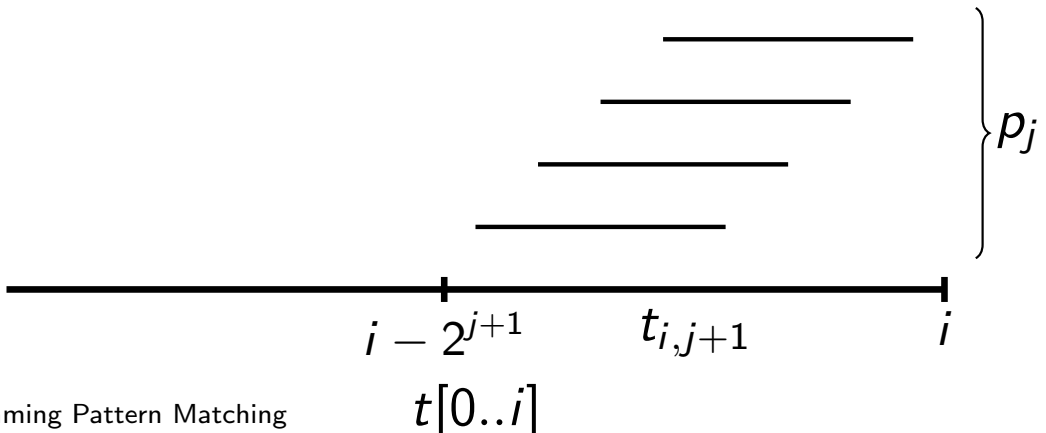


Streaming Pattern Matching, Main Idea

1. Let p_j be the prefix of the pattern of length 2^j.
   (We assume $p_k = p$ for $k = \lceil \log m \rceil$)
   Compute $h(p_i)$ when reading the pattern.



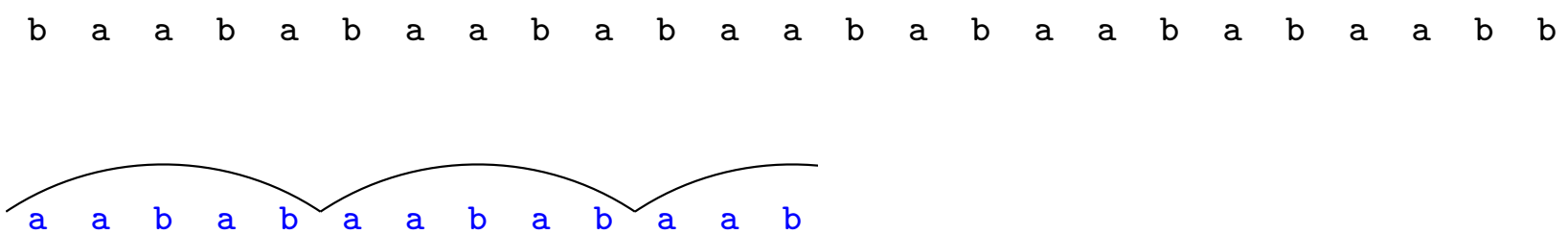
2. If i is a position in the text, let $t_{i,j}$ be the suffix of length $|p_j|$ of $t[0..i]$.
   Store (in a compact way) all the positions where p_j occurs in $t_{i,j+1}$.



Compact Representation of Occurrences

Fact from Combinatorics on Words

Assume that $|t| \leqslant 2|p|$ and that string p has at least three occurrences in string t. Then the occurrences of p in t form an arithmetic sequence with difference equal to the shortest period of p.



Compact Representation of Occurrences

Fact from Combinatorics on Words

Assume that $|t| \leqslant 2|p|$ and that string p has at least three occurrences in string t. Then the occurrences of p in t form an arithmetic sequence with difference equal to the shortest period of p.

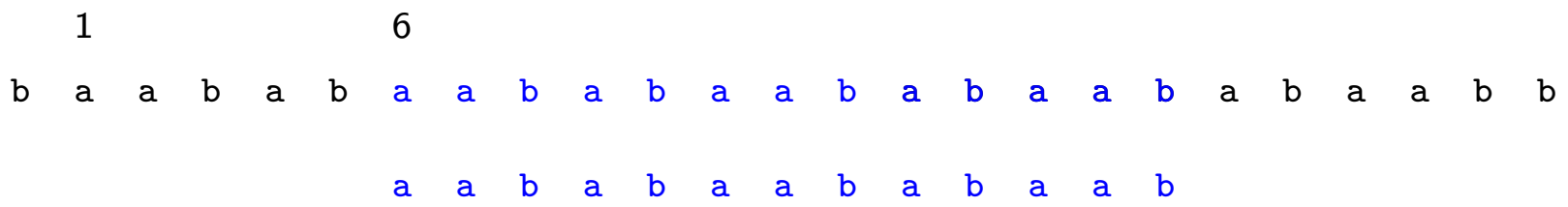
```
    1
b a a b a b a a b a b a a b a b a a b a b a a b b

  a a b a b a a b a b a a b
```

Compact Representation of Occurrences

Fact from Combinatorics on Words

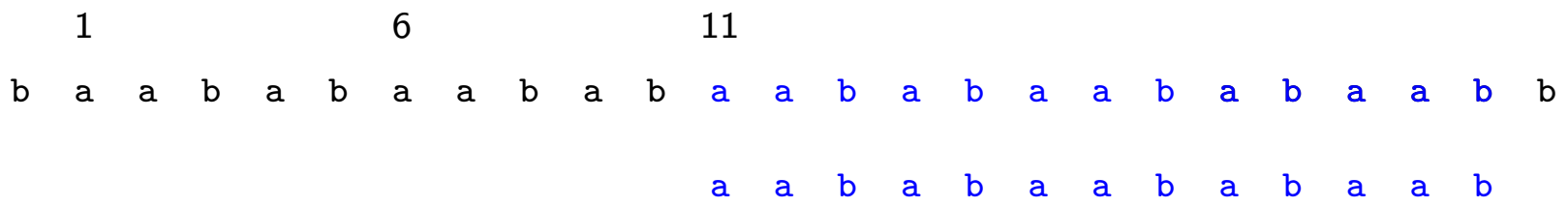
Assume that $|t| \leqslant 2|p|$ and that string p has at least three occurrences in string t. Then the occurrences of p in t form an arithmetic sequence with difference equal to the shortest period of p.



Compact Representation of Occurrences

Fact from Combinatorics on Words

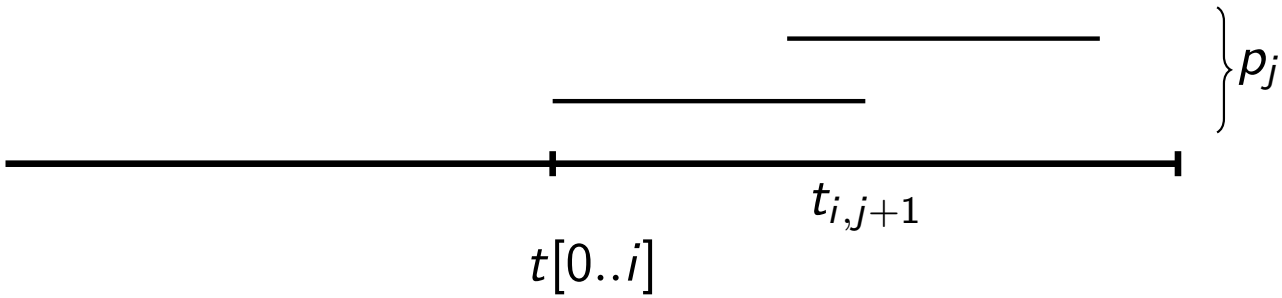
Assume that $|t| \leqslant 2|p|$ and that string p has at least three occurrences in string t. Then the occurrences of p in t form an arithmetic sequence with difference equal to the shortest period of p.



Streaming Pattern Matching

Compact representation of endpoints of occurrences of p_j in $t_{i,j+1}$ ($Occ(j)$):

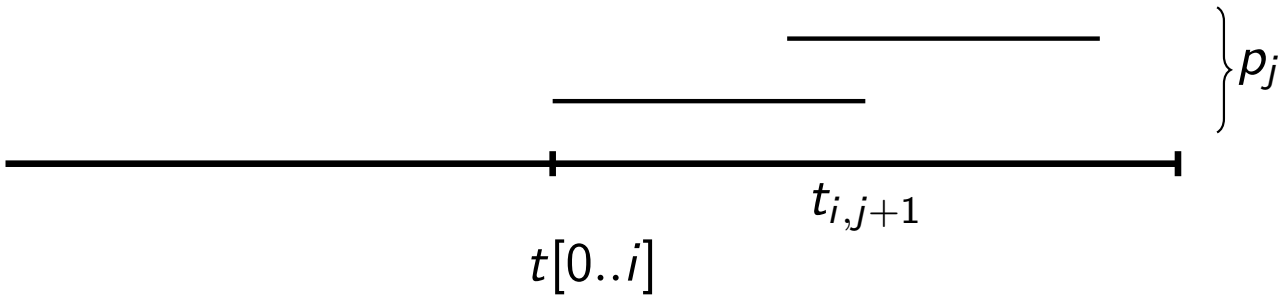
- If there are at most two occurrences, store their positions explicitly.



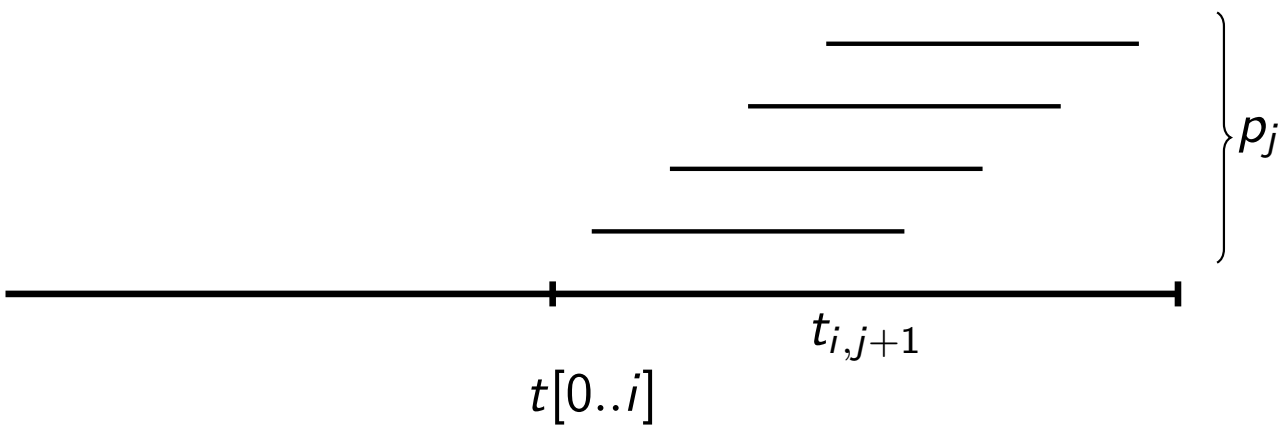
Streaming Pattern Matching

Compact representation of endpoints of occurrences of p_j in $t_{i,j+1}$ ($Occ(j)$):

- If there are at most two occurrences, store their positions explicitly.



- Otherwise the occurrences form an arithmetic progression: $a, a+b, a+2b, \ldots, a+(\ell-1)b$. It suffices to store: a, b, and ℓ.



Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j + 1)$

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0
$Occ(1)$	×
$Occ(2)$	×
$Occ(3)$	×

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0 0,1
$Occ(1)$	x
$Occ(2)$	x
$Occ(3)$	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1
$Occ(1)$	x	1
$Occ(2)$	x	
$Occ(3)$	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1
$Occ(1)$	x	1
$Occ(2)$	x	x
$Occ(3)$	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

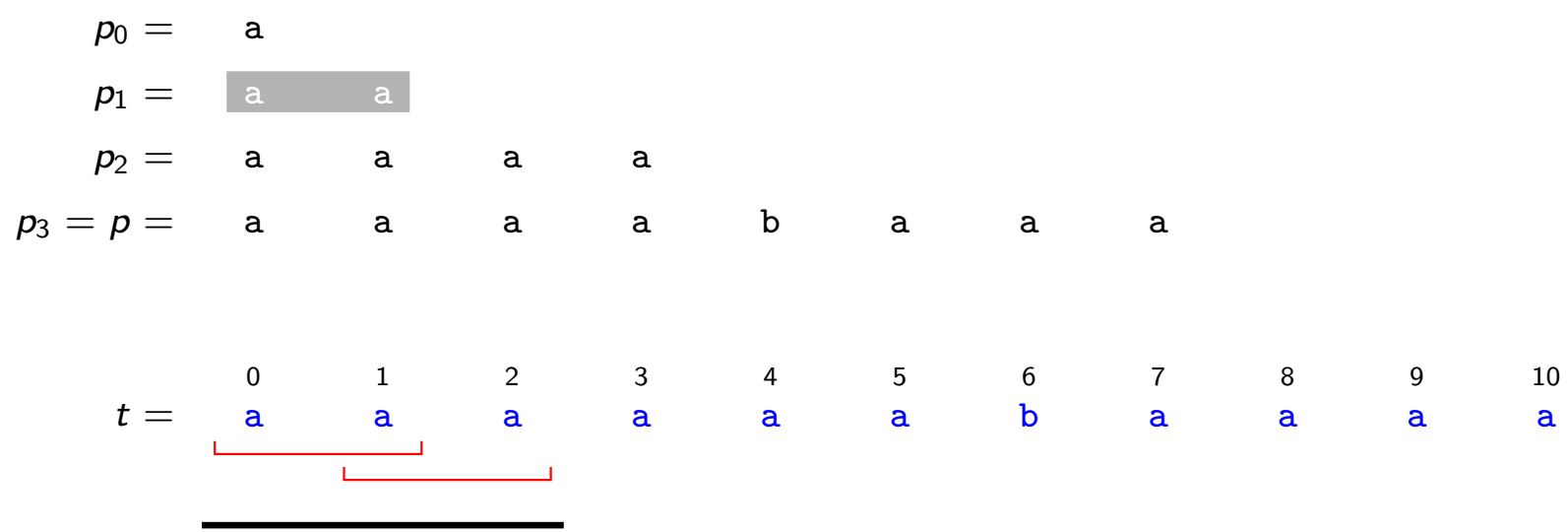
$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1 1,2
$Occ(1)$	x	1
$Occ(2)$	x	x
$Occ(3)$	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$



$Occ(0)$	0	0,1	1,2
$Occ(1)$	x	1	1,2
$Occ(2)$	x	x	
$Occ(3)$	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2
$Occ(1)$	x	1	1,2
$Occ(2)$	x	x	x
$Occ(3)$	x	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3
$Occ(1)$	x	1	1,2	
$Occ(2)$	x	x	x	
$Occ(3)$	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3
$Occ(1)$	x	1	1,2	1,2,3
$Occ(2)$	x	x	x	
$Occ(3)$	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3
$Occ(1)$	x	1	1,2	1,2,3
$Occ(2)$	x	x	x	3
$Occ(3)$	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3
$Occ(1)$	x	1	1,2	1,2,3
$Occ(2)$	x	x	x	3
$Occ(3)$	x	x	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

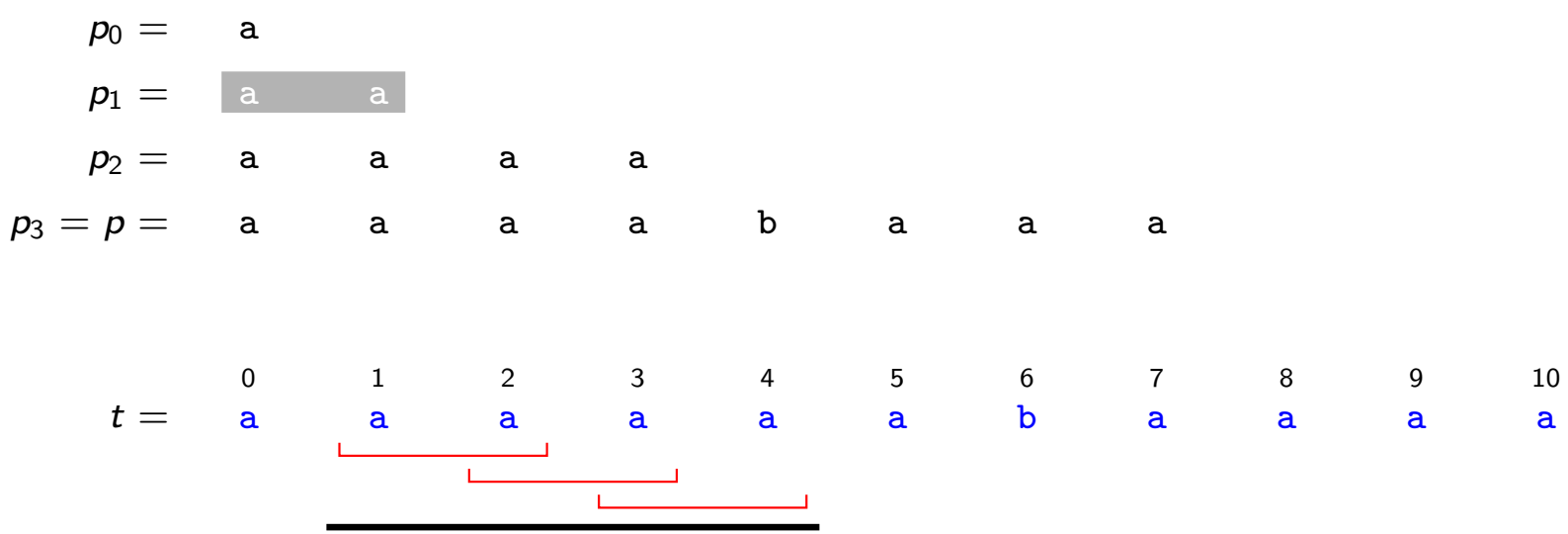
$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4
$Occ(1)$	x	1	1,2	1,2,3	
$Occ(2)$	x	x	x	3	
$Occ(3)$	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

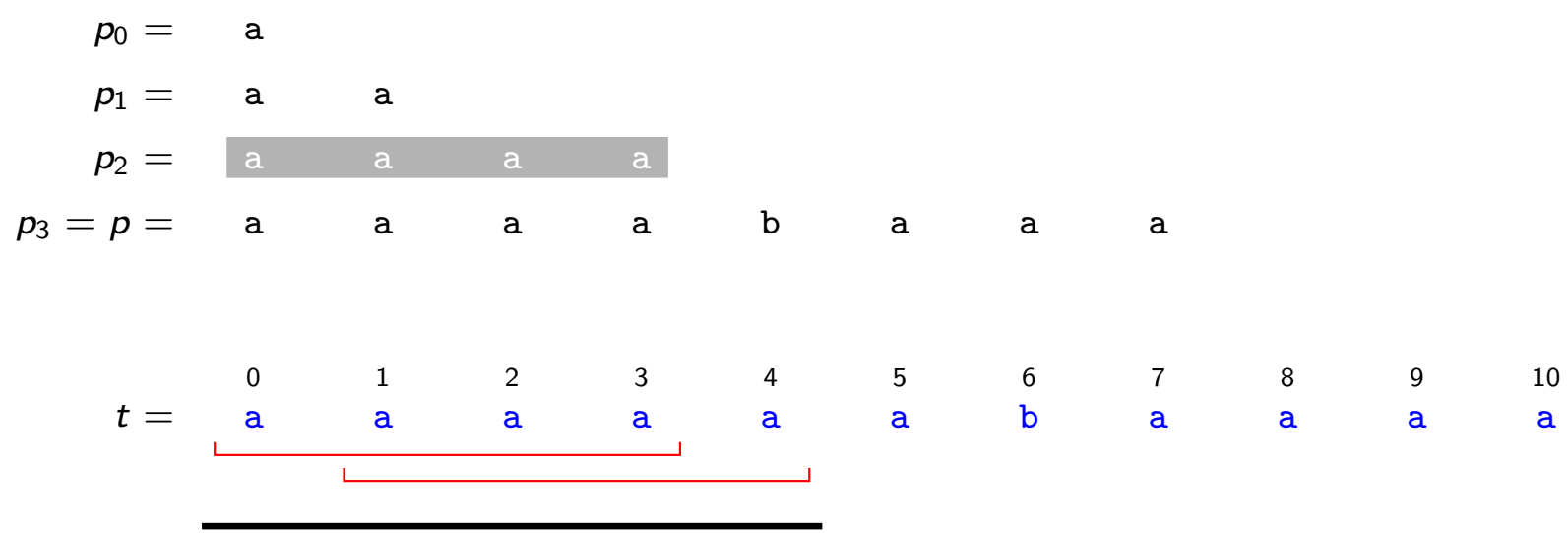


$Occ(0)$	0	0,1	1,2	2,3	3,4
$Occ(1)$	x	1	1,2	1,2,3	2,3,4
$Occ(2)$	x	x	x	3	
$Occ(3)$	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$



$Occ(0)$	0	0,1	1,2	2,3	3,4
$Occ(1)$	x	1	1,2	1,2,3	2,3,4
$Occ(2)$	x	x	x	3	3,4
$Occ(3)$	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4
$Occ(1)$	x	1	1,2	1,2,3	2,3,4
$Occ(2)$	x	x	x	3	3,4
$Occ(3)$	x	x	x	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	
$Occ(2)$	x	x	x	3	3,4	
$Occ(3)$	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

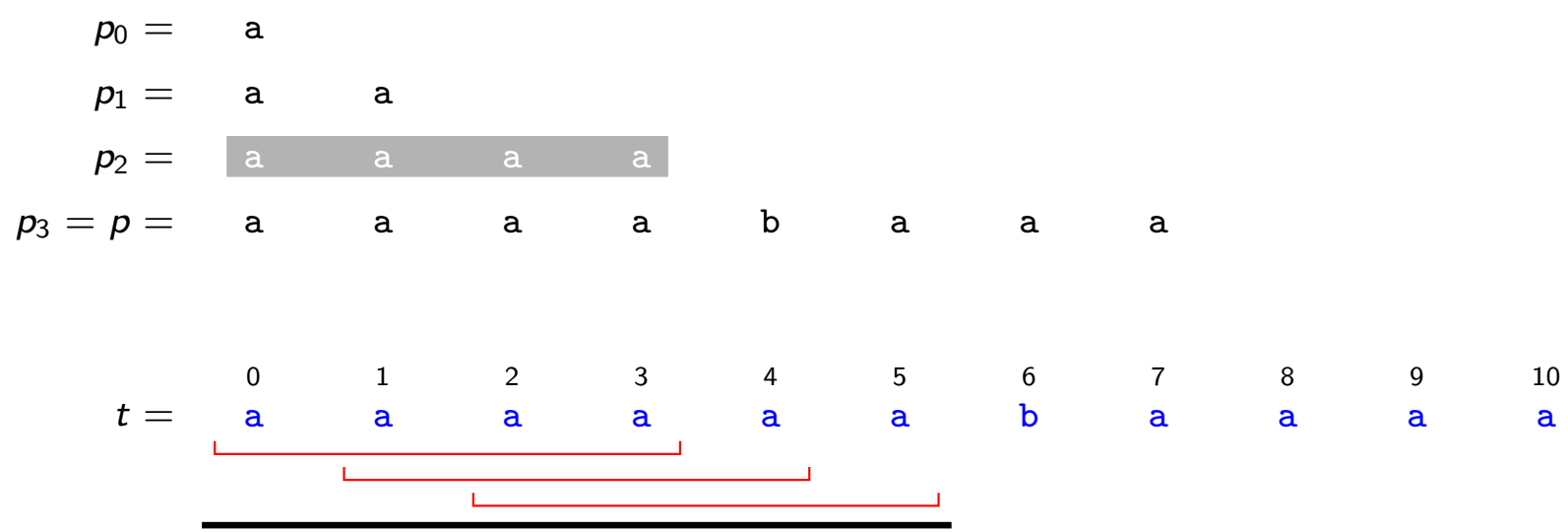
1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5					
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5					
$Occ(2)$	x	x	x	3	3,4						
$Occ(3)$	x	x	x	x	x						

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$



$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5
$Occ(2)$	x	x	x	3	3,4	3,4,5
$Occ(3)$	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5
$Occ(2)$	x	x	x	3	3,4	3,4,5
$Occ(3)$	x	x	x	x	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

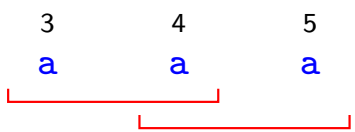
$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5				
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5					
$Occ(2)$	x	x	x	3	3,4	3,4,5					
$Occ(3)$	x	x	x	x	x	x					

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a



$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5
$Occ(2)$	x	x	x	3	3,4	3,4,5	
$Occ(3)$	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5
$Occ(3)$	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5				
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5				
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5				
$Occ(3)$	x	x	x	x	x	x	x				

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	
$Occ(3)$	x	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	
$Occ(3)$	x	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5
$Occ(3)$	x	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5
$Occ(3)$	x	x	x	x	x	x	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8		
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5			
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5			
$Occ(3)$	x	x	x	x	x	x	x	x			

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

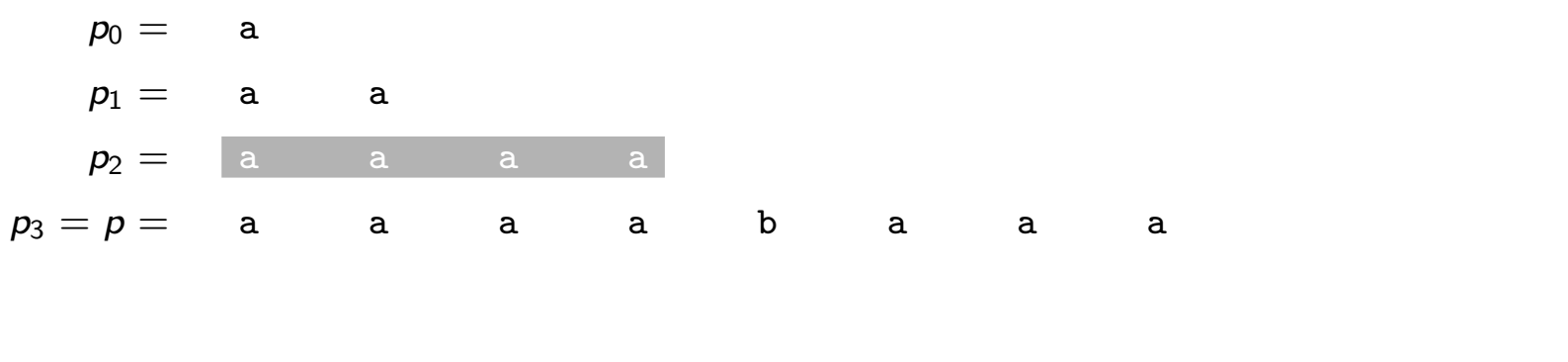
1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8		
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	8		
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5			
$Occ(3)$	x	x	x	x	x	x	x	x			

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$



$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5
$Occ(3)$	x	x	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5
$Occ(3)$	x	x	x	x	x	x	x	x	x

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8	8,9	
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7		
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5		
$Occ(3)$	x	x	x	x	x	x	x	x	x		

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a									
$p_1 =$	a	a								
$p_2 =$	a	a	a	a						
$p_3 = p =$	a	a	a	a	b	a	a	a		
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a
$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8	8,9	
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7	8,9	
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5		
$Occ(3)$	x	x	x	x	x	x	x	x	x		

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a										
$p_1 =$	a	a									
$p_2 =$	a	a	a	a							
$p_3 = p =$	a	a	a	a	b	a	a	a			
	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8	8,9
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7	8,9
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5	5
$Occ(3)$	x	x	x	x	x	x	x	x	x	

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

$p_0 =$	a								
$p_1 =$	a	a							
$p_2 =$	a	a	a	a					
$p_3 = p =$	a	a	a	a	b	a	a	a	

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8	8,9
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7	8,9
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5	5
$Occ(3)$	x	x	x	x	x	x	x	x	x	9

Streaming Pattern Matching

First update $Occ(0)$. Then update $Occ(j)$, for each j:

1. Remove the first element of $Occ(j)$ if it is equal to $i' = i - 2^j$
2. If the first element of $Occ(j)$ is $i' + 1$, check if it can be extended into an occurrence of p_{j+1} and, if so, add it to $Occ(j+1)$

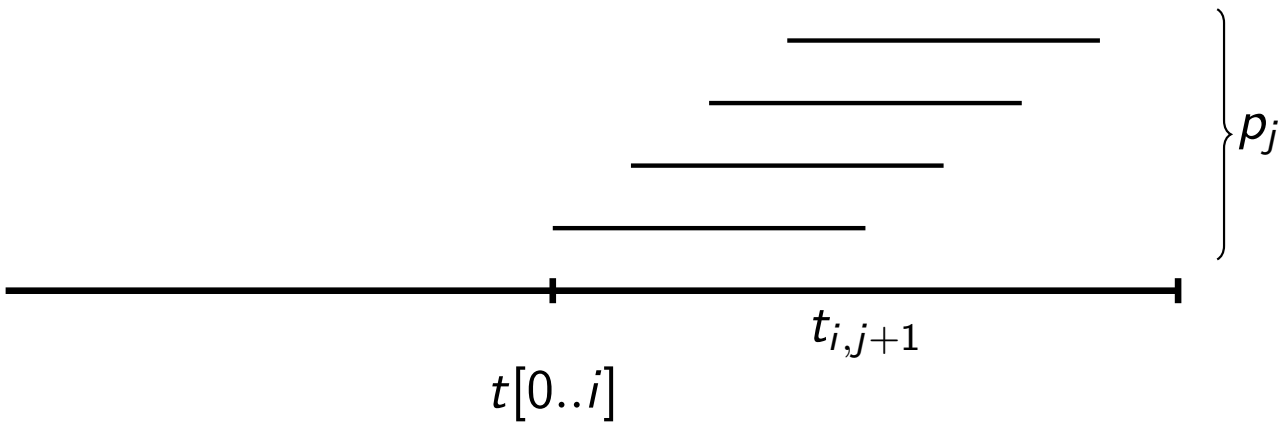
$p_0 =$	a								
$p_1 =$	a	a							
$p_2 =$	a	a	a	a					
$p_3 = p =$	a	a	a	a	b	a	a	a	

	0	1	2	3	4	5	6	7	8	9	10
$t =$	a	a	a	a	a	a	b	a	a	a	a

$Occ(0)$	0	0,1	1,2	2,3	3,4	4,5	5	7	7,8	8,9
$Occ(1)$	x	1	1,2	1,2,3	2,3,4	3,4,5	4,5	5	7	8,9
$Occ(2)$	x	x	x	3	3,4	3,4,5	3,4,5	3,4,5	4,5	5
$Occ(3)$	x	x	x	x	x	x	x	x	x	9

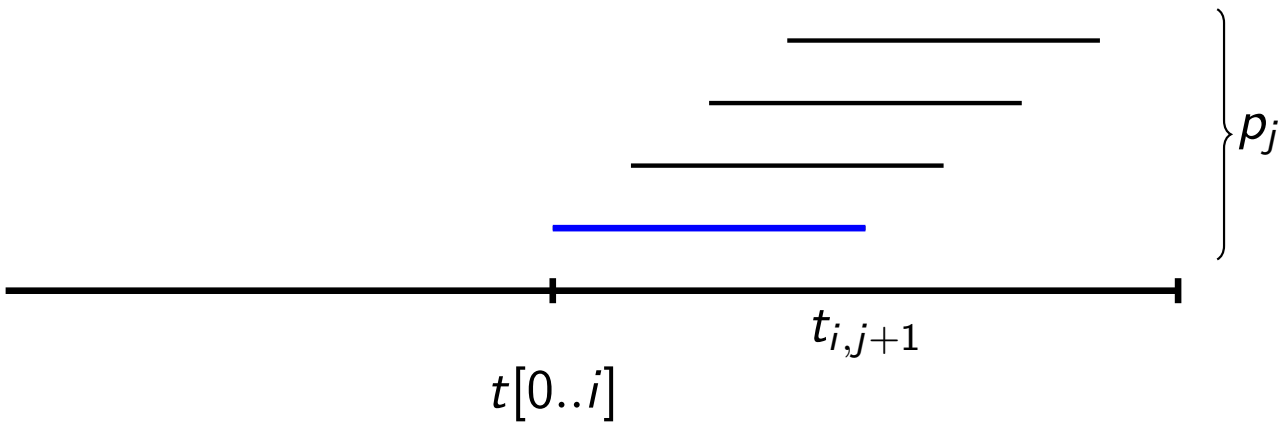
Streaming Pattern Matching

Situation 1: When an occurrence of p_j can be extended to an occurrence of p_{j+1}?



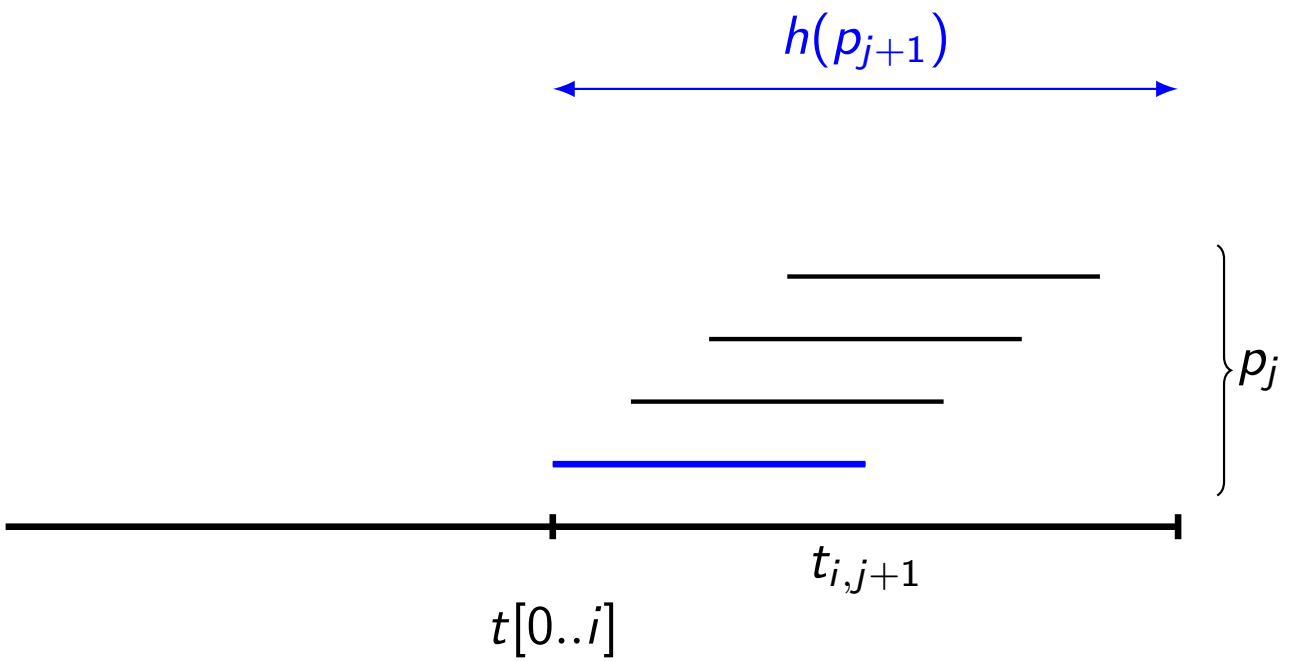
Streaming Pattern Matching

Situation 1: When an occurrence of p_j can be extended to an occurrence of p_{j+1}?



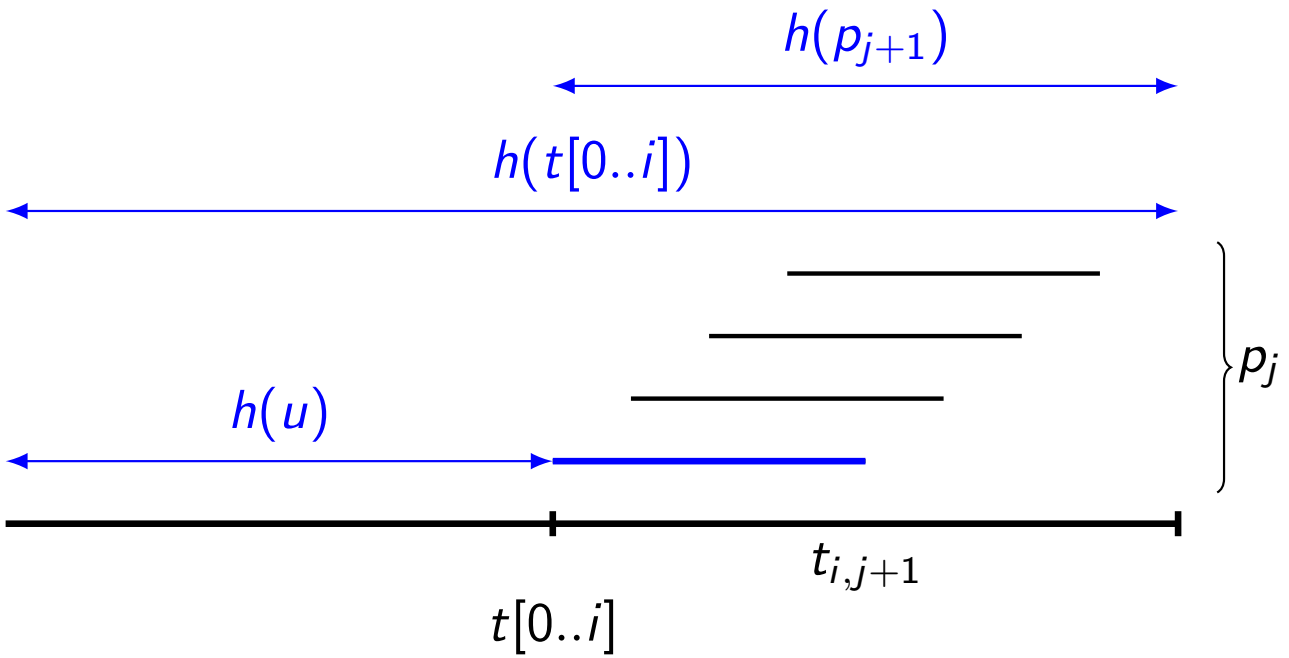
Streaming Pattern Matching

Situation 1: When an occurrence of p_j can be extended to an occurrence of p_{j+1}?



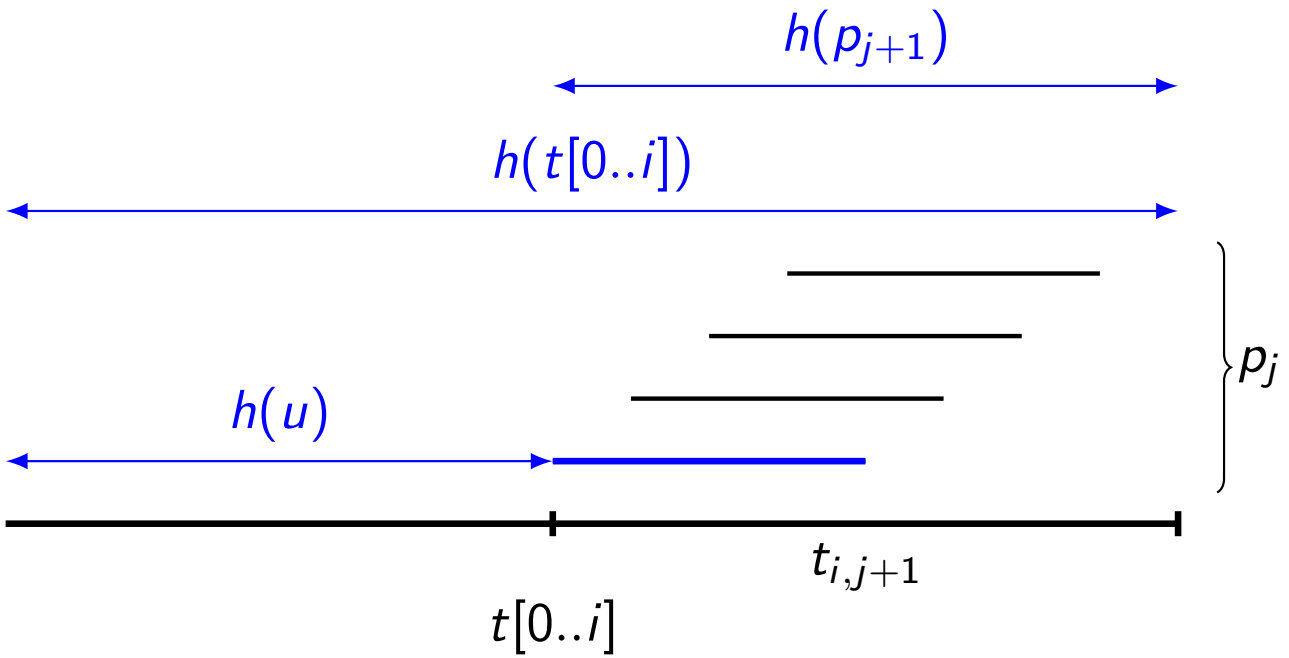
Streaming Pattern Matching

Situation 1: When an occurrence of p_j can be extended to an occurrence of p_{j+1}?



Streaming Pattern Matching

Situation 1: When an occurrence of p_j can be extended to an occurrence of p_{j+1}?

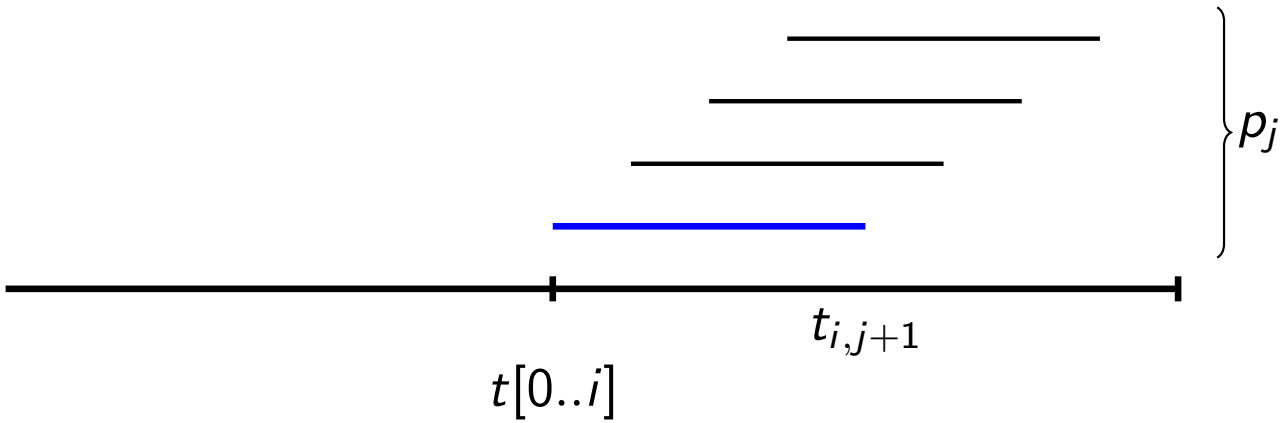


Stored data:

- $h(t[0..i])$
- The hash from $t[0]$ until the first occurrence of p_j in $t_{i,j+1}$

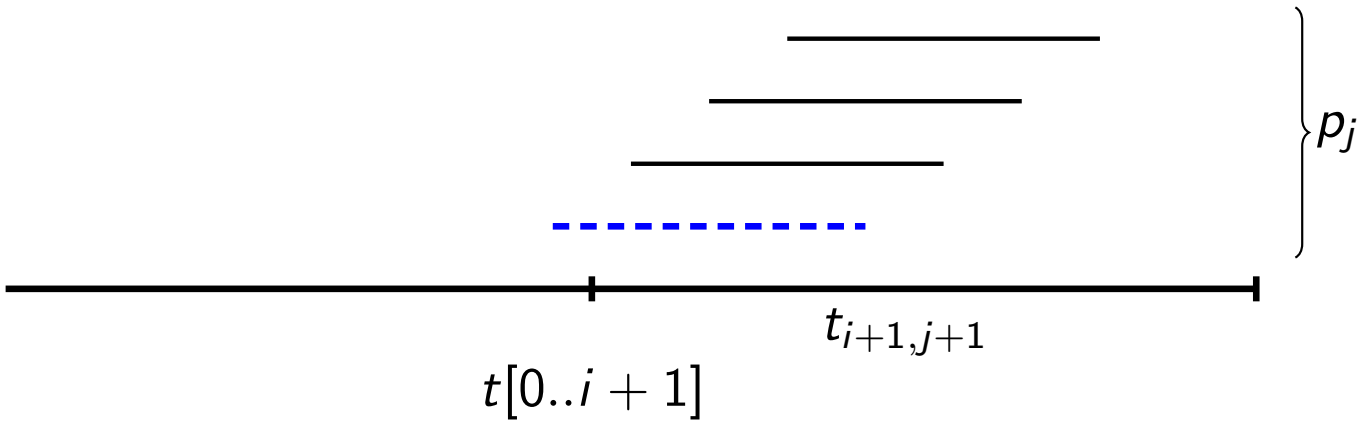
Streaming Pattern Matching

Situation 2: How to handle deletion of the first stored occurrence of p_j?



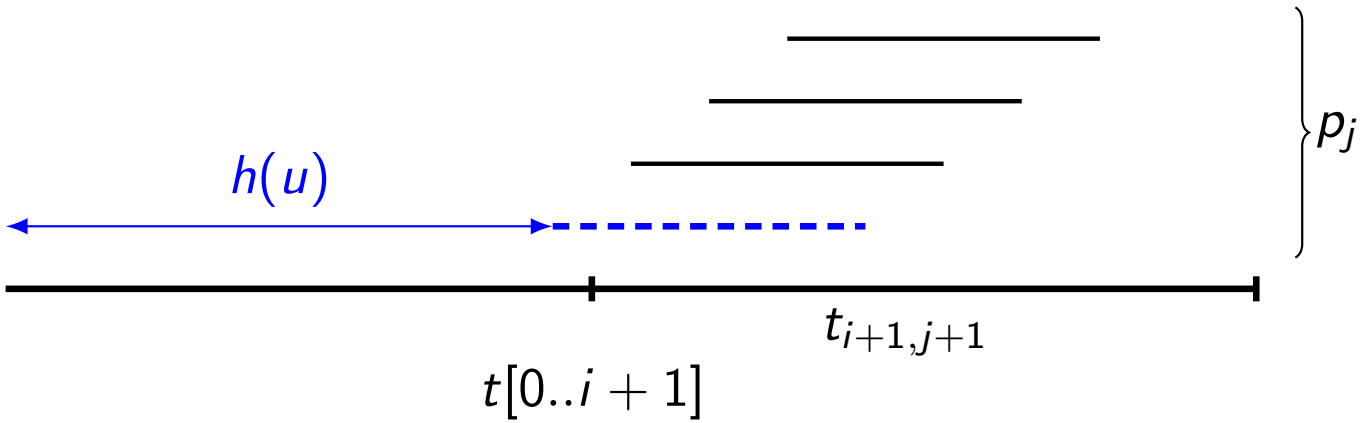
Streaming Pattern Matching

Situation 2: How to handle deletion of the first stored occurrence of p_j?



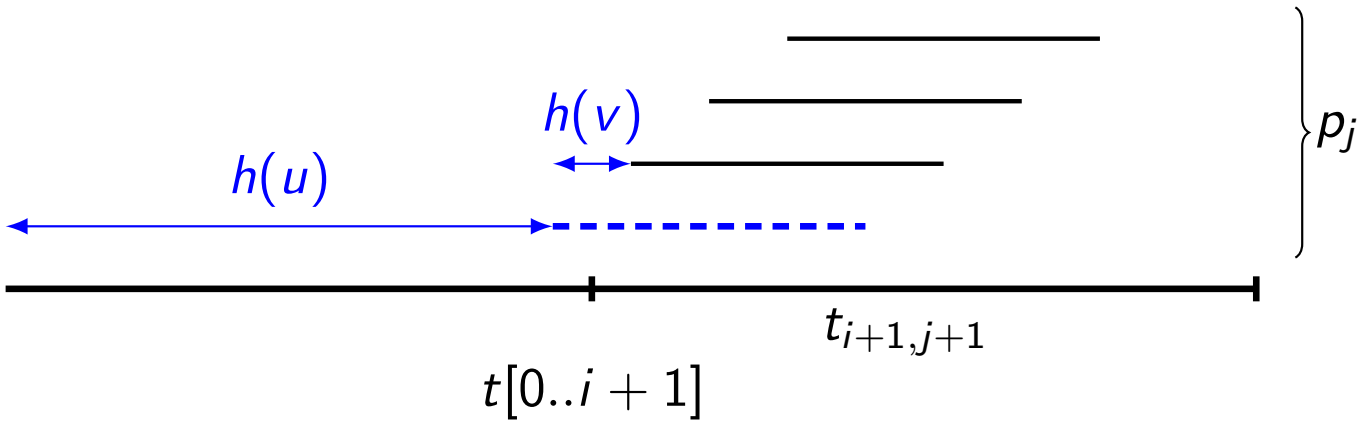
Streaming Pattern Matching

Situation 2: How to handle deletion of the first stored occurrence of p_j?



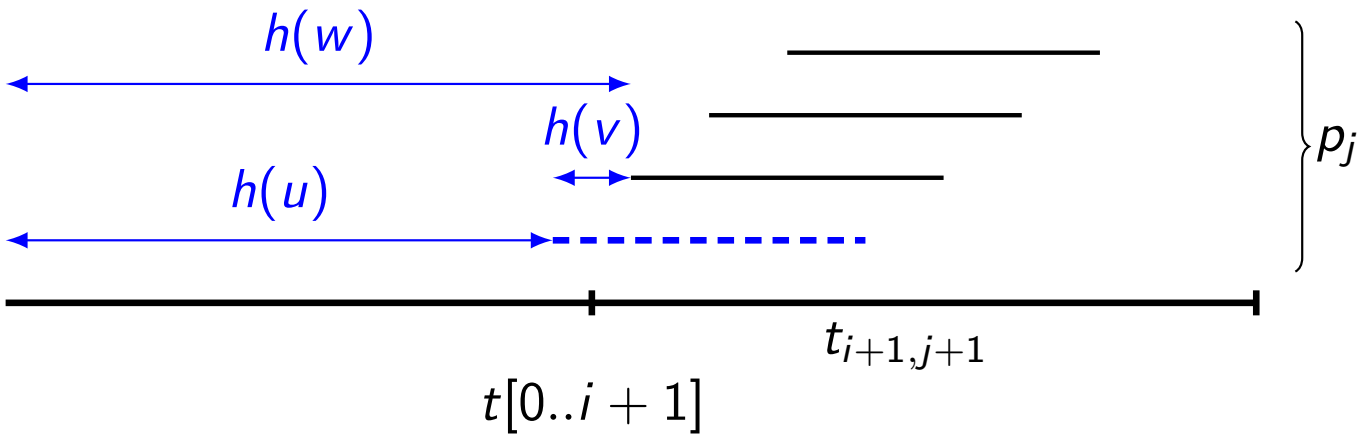
Streaming Pattern Matching

Situation 2: How to handle deletion of the first stored occurrence of p_j?



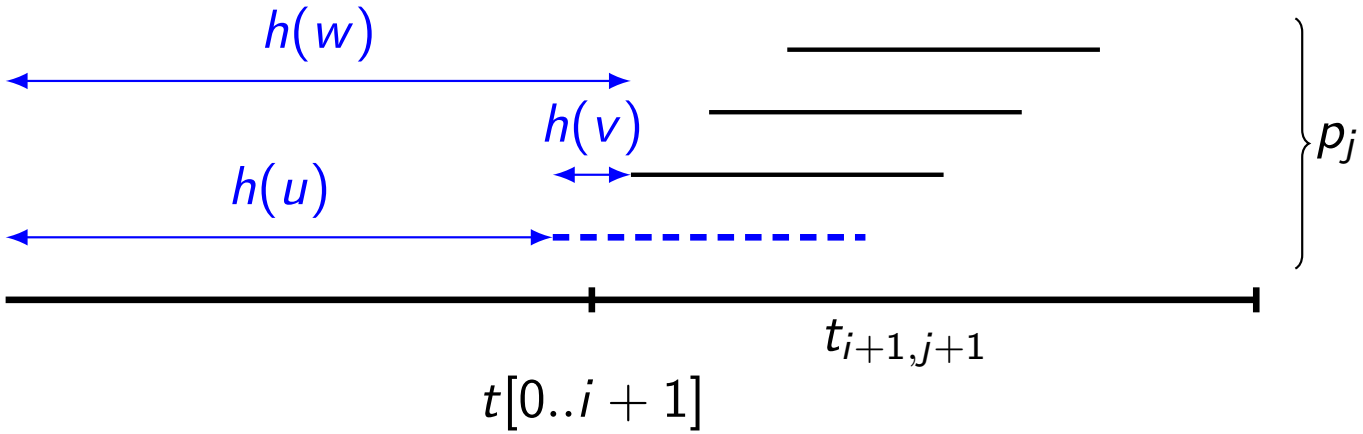
Streaming Pattern Matching

Situation 2: How to handle deletion of the first stored occurrence of p_j?



Streaming Pattern Matching

Situation 2: How to handle deletion of the first stored occurrence of p_j?

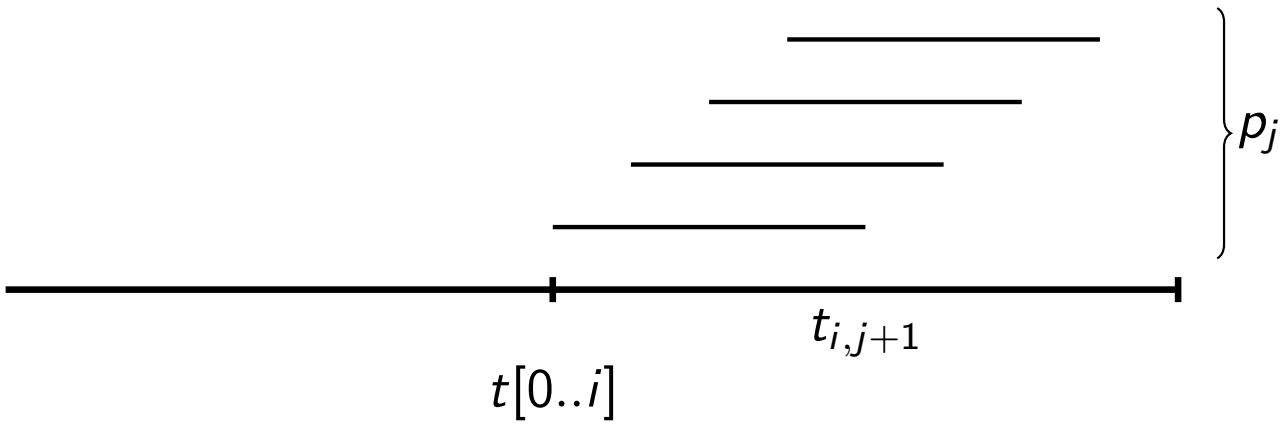


Stored data:

- The hash of the period of p_j (with special handling of the cases of 0, 1, or 2 occurrences of p_j in $t_{i,j+1}$)

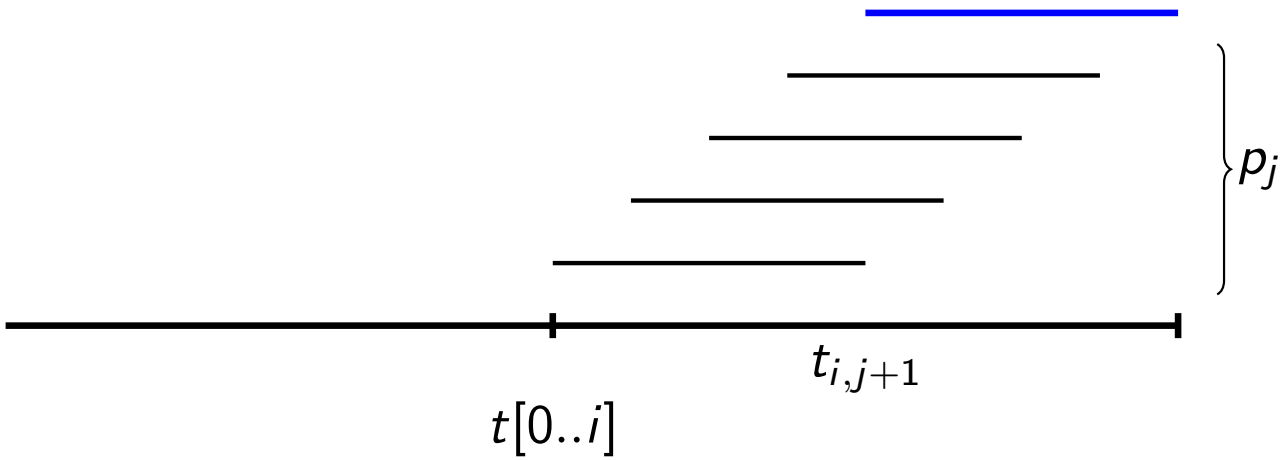
Streaming Pattern Matching

Situation 3: How to compute the desired hashes on a new occurrence of p_j?



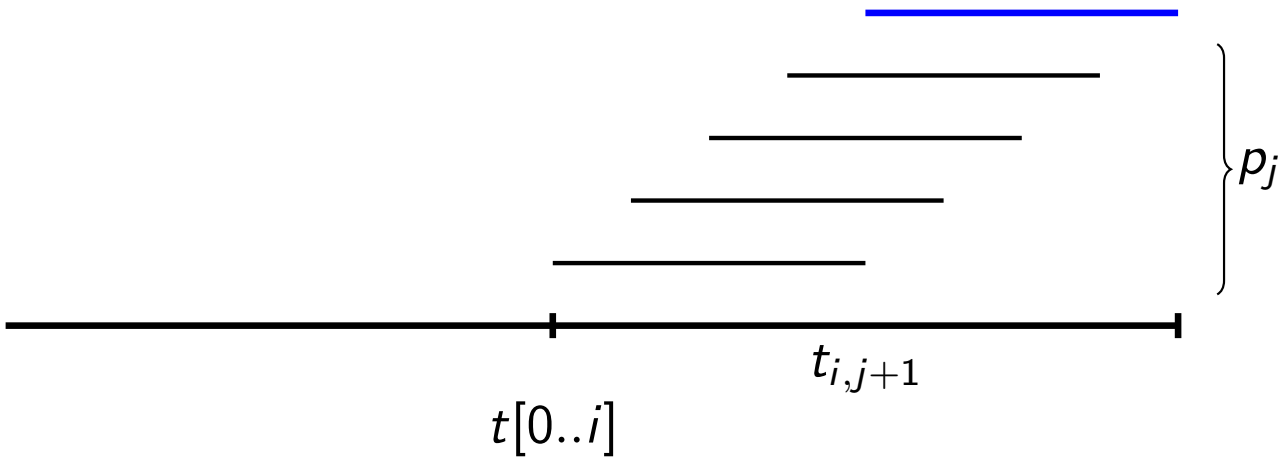
Streaming Pattern Matching

Situation 3: How to compute the desired hashes on a new occurrence of p_j?



Streaming Pattern Matching

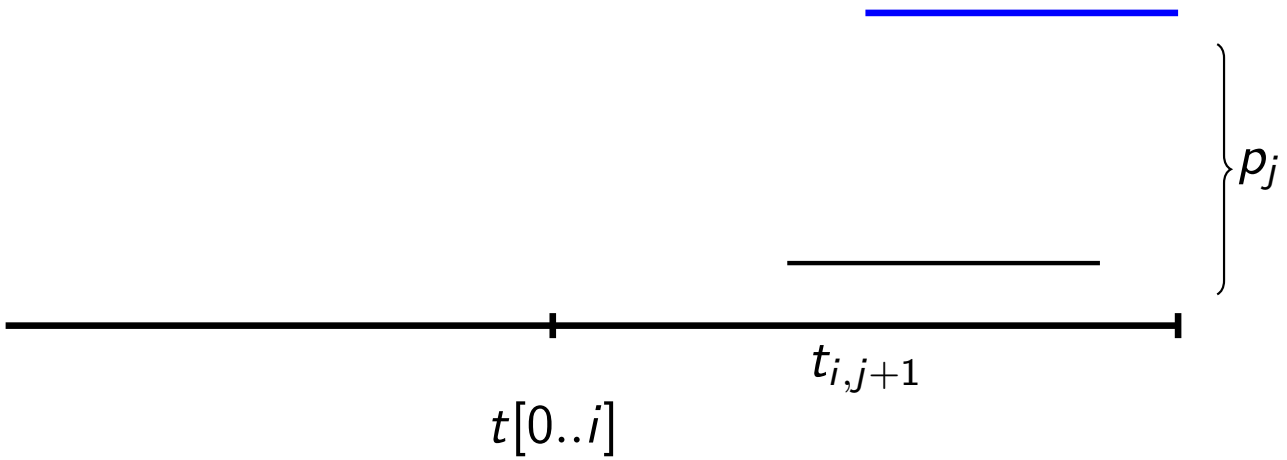
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing

Streaming Pattern Matching

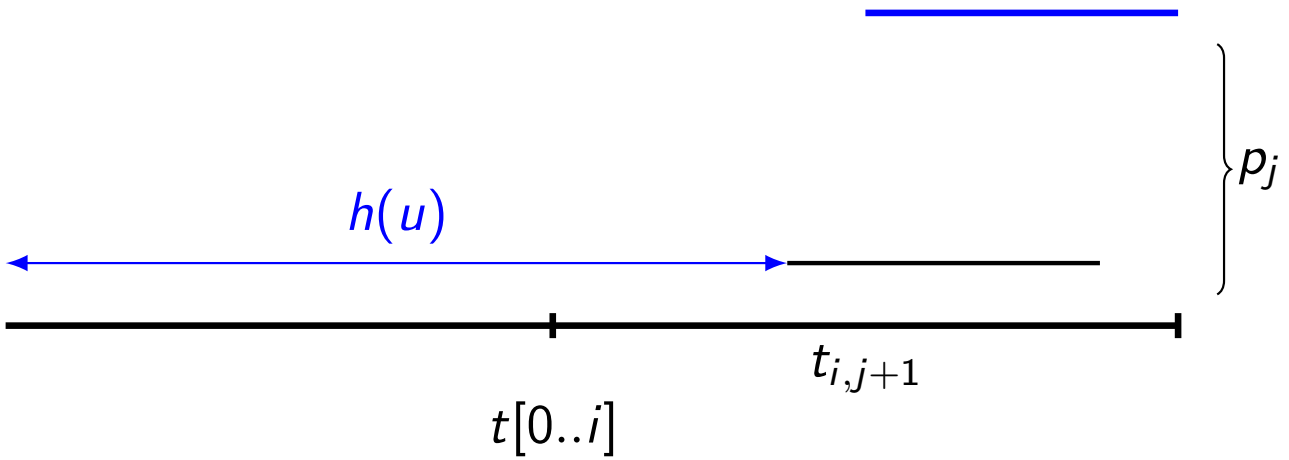
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing

Streaming Pattern Matching

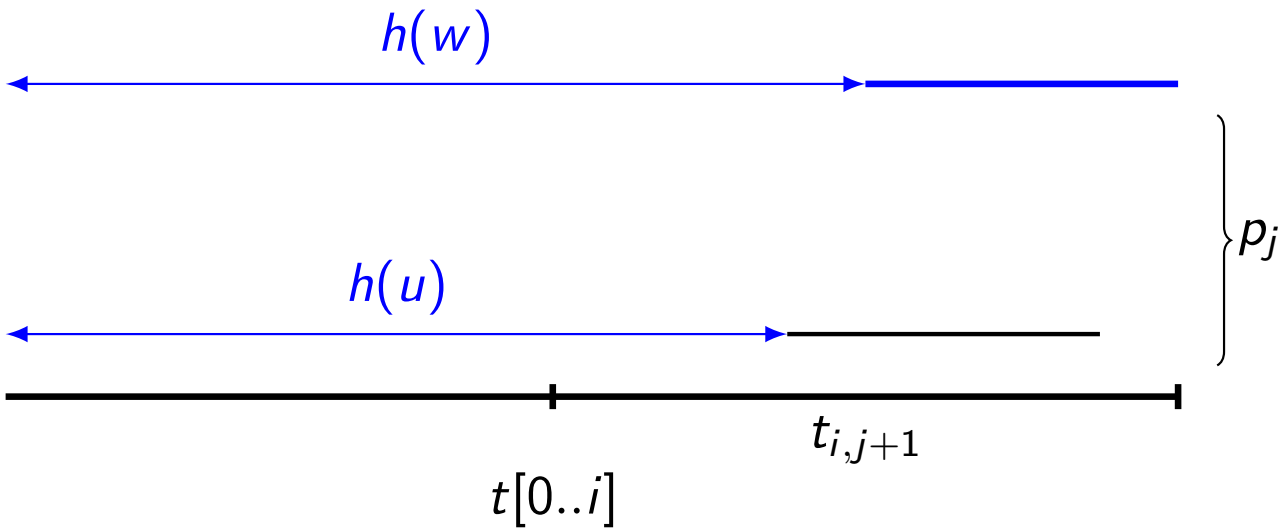
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing

Streaming Pattern Matching

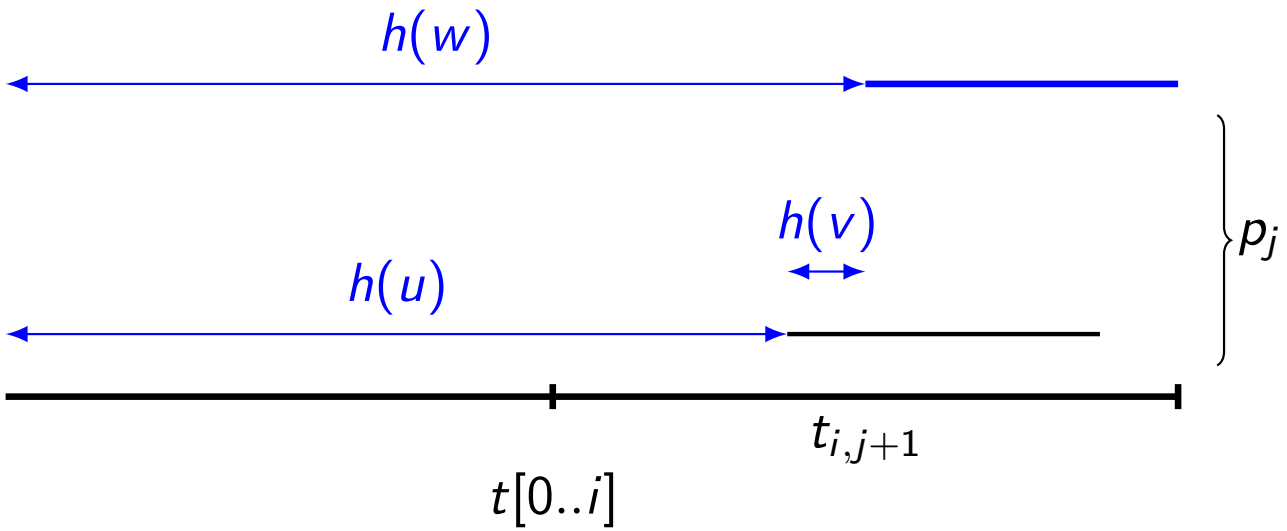
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing

Streaming Pattern Matching

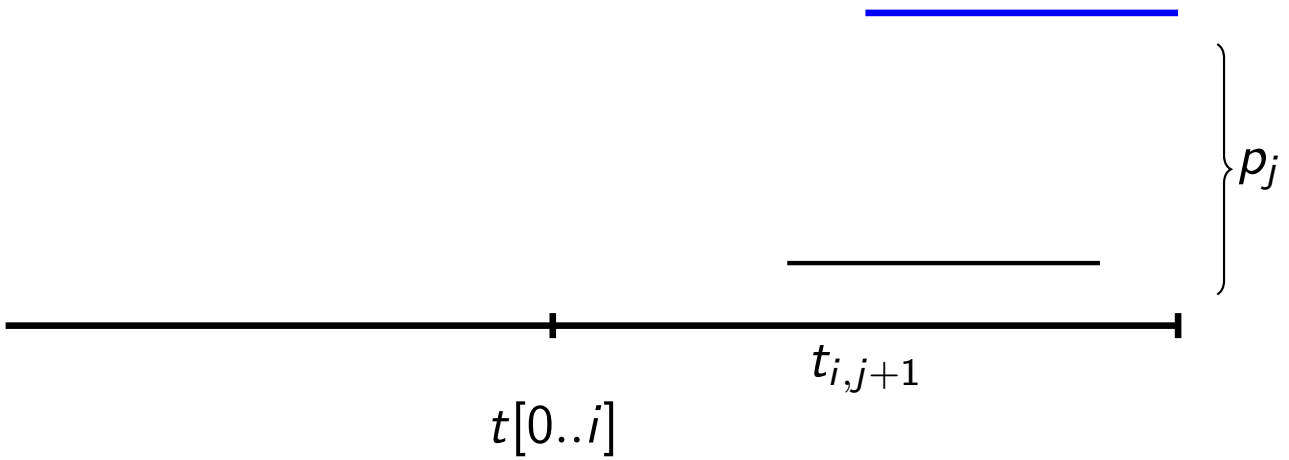
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing

Streaming Pattern Matching

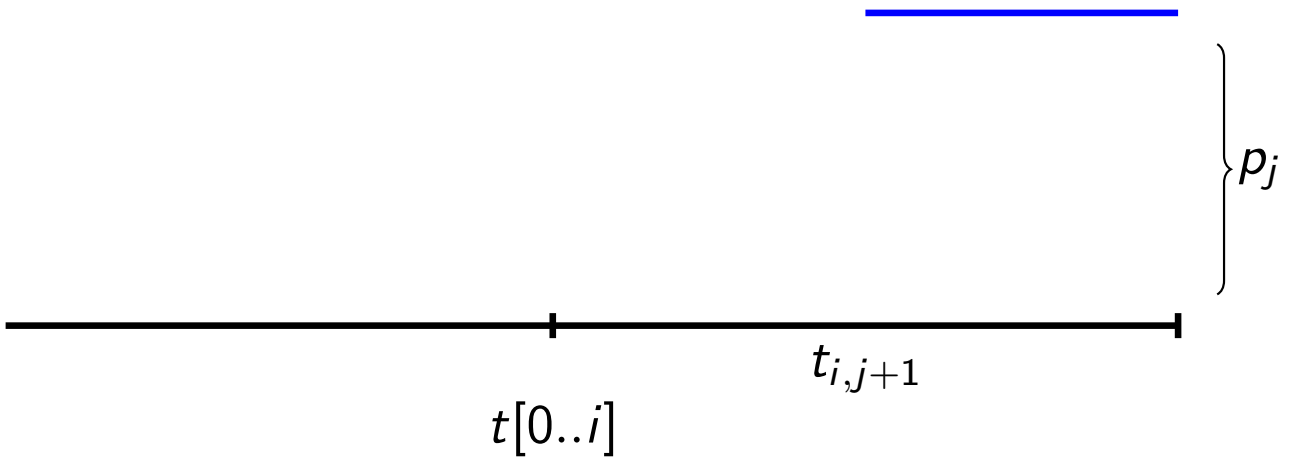
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing
- Or compute the hash of the period of p_j

Streaming Pattern Matching

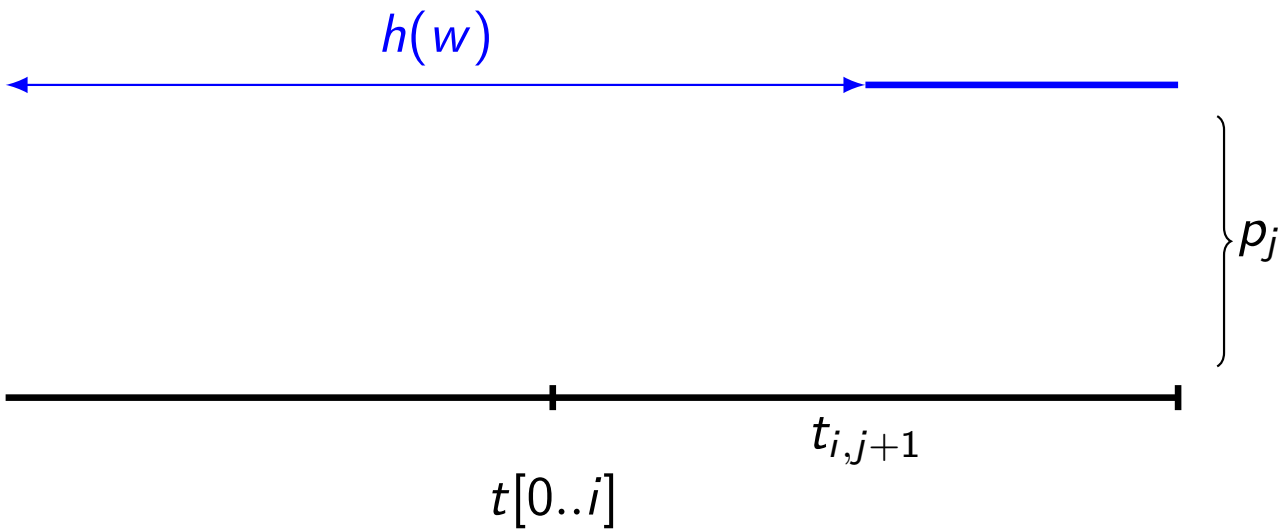
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing
- Or compute the hash of the period of p_j

Streaming Pattern Matching

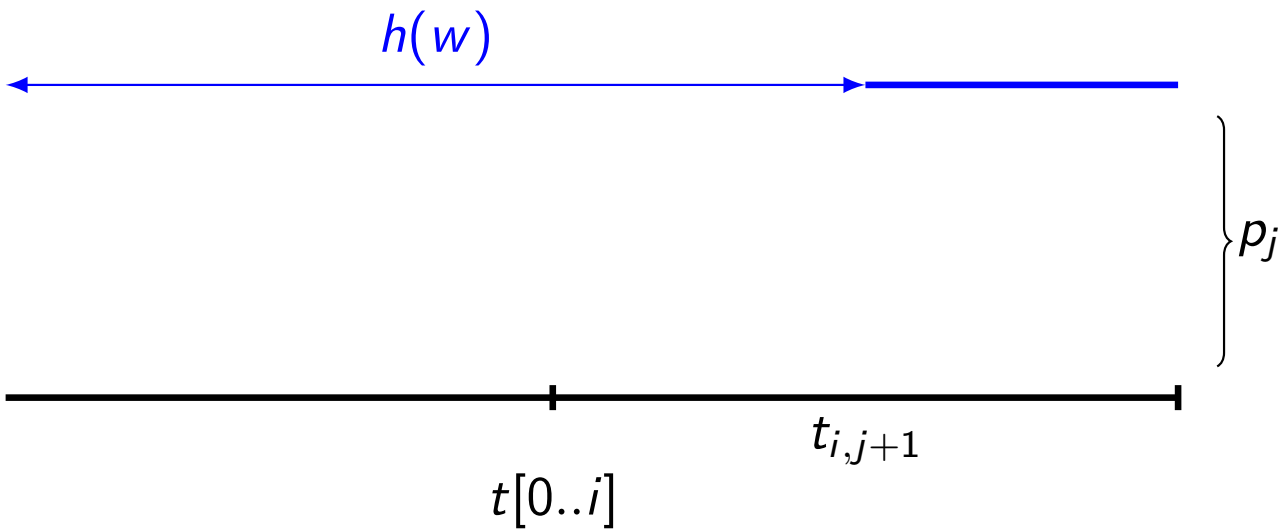
Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing
- Or compute the hash of the period of p_j

Streaming Pattern Matching

Situation 3: How to compute the desired hashes on a new occurrence of p_j?



- Either do nothing
- Or compute the hash of the period of p_j
- Or set the hash from $t[0]$ until the first occurrence of p_j in $t_{i,j+1}$

Summary

Stored data:

- $h(t[0..i])$

And for each $j = 0, \ldots, \lceil \log m \rceil$:

- a, b, ℓ of the progression
- $h(p_j)$
- The hash from $t[0]$ until the first occurrence of p_j in $t_{i,j+1}$
- The hash of the period of p_j (with special handling of the cases of 0, 1, or 2 occurrences of p_j in $t_{i,j+1}$)

We make $O(1)$ updates for each j when advancing i.

Summary

Stored data:

- $h(t[0..i])$

And for each $j = 0, \ldots, \lceil \log m \rceil$:

- a, b, ℓ of the progression
- $h(p_j)$
- The hash from $t[0]$ until the first occurrence of p_j in $t_{i,j+1}$
- The hash of the period of p_j (with special handling of the cases of 0, 1, or 2 occurrences of p_j in $t_{i,j+1}$)

We make $O(1)$ updates for each j when advancing i.

Theorem

Pattern matching can be solved in the streaming model with $O(\log m)$ space and $O(\log m)$ time per arriving symbol. The algorithm may incur small probability errors.

References

First Algorithm

B. Porat, E. Porat, Exact and approximate pattern matching in the streaming model. *Proceedings of the 50th Annual Symposium on Foundations of Computer Science*, FOCS'09 (2009)

$O(\log m)$ *space,* $O(\log m)$ *time per symbol*

Better Algorithm

D. Breslauer, Z. Galil, Real-time streaming string-matching. *Proceedings of the 22nd Annual Symposium on Combinatorial Pattern Matching*, CPM'11 (2011)

$O(\log m)$ *space,* $O(1)$ *time per symbol;*
or $O(\log m)$ *time per symbol with a simpler algorithm*

References

What if up to k mismatches are allowed in occurrence?

First Algorithm

B. Porat, E. Porat, FOCS'09

$O(k^3 \operatorname{polylog} n)$ *space,* $O(k^2 \operatorname{polylog} n)$ *time per symbol*

Better Algorithm

R. Clifford et al., The k-mismatch problem revisited. *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'16 (2016)

$O(k^2 \operatorname{polylog} n)$ *space,* $O(\sqrt{k} \operatorname{polylog} n)$ *time per symbol*

Most Recent Algorithm

R. Clifford, **T. Kociumaka**, E. Porat, The streaming k-mismatch problem, *30th ACM-SIAM Symposium on Discrete Algorithms*, SODA'19 (2019)

$O(k \operatorname{polylog} n)$ *space and* $O(\sqrt{k} \operatorname{polylog} n)$ *time per symbol*