PSC 2024

The Praga Stringology Conference

Prague
Czech Republic
August 26-27, 2024

Beyond Horspool: A Comparative Analysis in Sampled Matching

Simone Faro[1], Francesco Pio Marino[1,2], Andrea Moschetto[1]

[1] Dipartimento di Matematica e Informatica, Università di Catania, viale A.Doria n.6, 95125, Catania, Italia

[2] Univ Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie, Normandie Univ, LITIS UR 4108, CNRS NormaSTIC FR 3638, IRIB, Rouen F-76000, France

PSC 2024

The Praga Stringology Conference

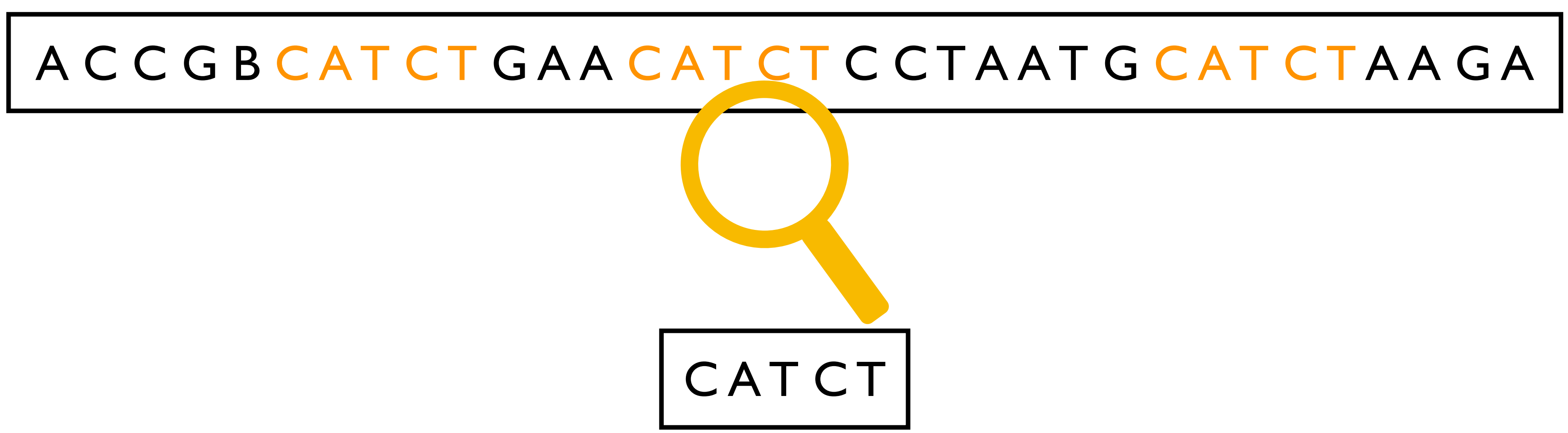
Prague
Czech Republic
August 26-27, 2024

Beyond Horspool: A Comparative Analysis in Sampled Matching

Simone Faro[1], Francesco Pio Marino[1,2], Andrea Moschetto[1]

[1] Dipartimento di Matematica e Informatica, Università di Catania, viale A.Doria n.6, 95125, Catania, Italia

[2] Univ Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie, Normandie Univ, LITIS UR 4108, CNRS NormaSTIC FR 3638, IRIB, Rouen F-76000, France



PSC 2024

The Praga Stringology Conference

Prague
Czech Republic
August 26-27, 2024

Beyond Horspool: A Comparative Analysis in Sampled Matching

Simone Faro[1], Francesco Pio Marino[1,2], Andrea Moschetto[1]

[1] Dipartimento di Matematica e Informatica, Università di Catania, viale A.Doria n.6, 95125, Catania, Italia

[2] Univ Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie, Normandie Univ, LITIS UR 4108, CNRS NormaSTIC FR 3638, IRIB, Rouen F-76000, France

Abstract. The *exact online string matching* problem, pivotal in fields ranging from computational biology to data compression, involves identifying all instances of a specified pattern within a text. Despite extensive examination over the decades, this problem has remained computationally challenging due to the time and space limitations inherent in traditional online and offline methods, respectively. Introduced in 1991, *sampled string matching* has now emerged as a groundbreaking approach, ingeniously combining classical online string matching techniques with efficient text sampling methods. This approach not only addresses the spatial constraints of indexed string matching but also significantly reduces the search duration in online environments, achieving speed increases of up to hundreds of times while requiring less than 4% of the text size for its partial index. In this paper, we explore the adaptability of various online string matching algorithms within the framework of sampled string matching, which has traditionally relied on the Horspool algorithm. Our investigation reveals that integrating alternative string matching algorithms as subroutines markedly enhances overall performance. These findings highlight the potential for reevaluating established methodologies in light of newer, more dynamic solutions and set the stage for transformative impacts across multiple domains.

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

1977

- Knuth, Morris and Pratt (KMP)
- Boyer and Moore (BM)

Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

1977

- Knuth, Morris and Pratt (KMP)
- Boyer and Moore (BM)
- $\mathcal{O}(n \log_\sigma(m)/m)$ time complexity on the average Yao

Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

1977

- Knuth, Morris and Pratt (KMP)
- Boyer and Moore (BM)
- $\mathcal{O}(n \log_\sigma(m)/m)$ time complexity on the average Yao

1994

- Backward-DAWG-Matching (BDM) Crochemore and Rytter

Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

1977

- Knuth, Morris and Pratt (KMP)
- Boyer and Moore (BM)
- $\mathcal{O}(n \log_\sigma(m)/m)$ time complexity on the average Yao

1994

- Backward-DAWG-Matching (BDM) Crochemore and Rytter

2013

- Faro and Lecroq (ACM C.Survey)

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

180 Algorithms

1977

- Knuth, Morris and Pratt (KMP)
- Boyer and Moore (BM)
- $\mathcal{O}(n \log_\sigma(m)/m)$ time complexity on the average Yao

1994

- Backward-DAWG-Matching (BDM) Crochemore and Rytter

2013

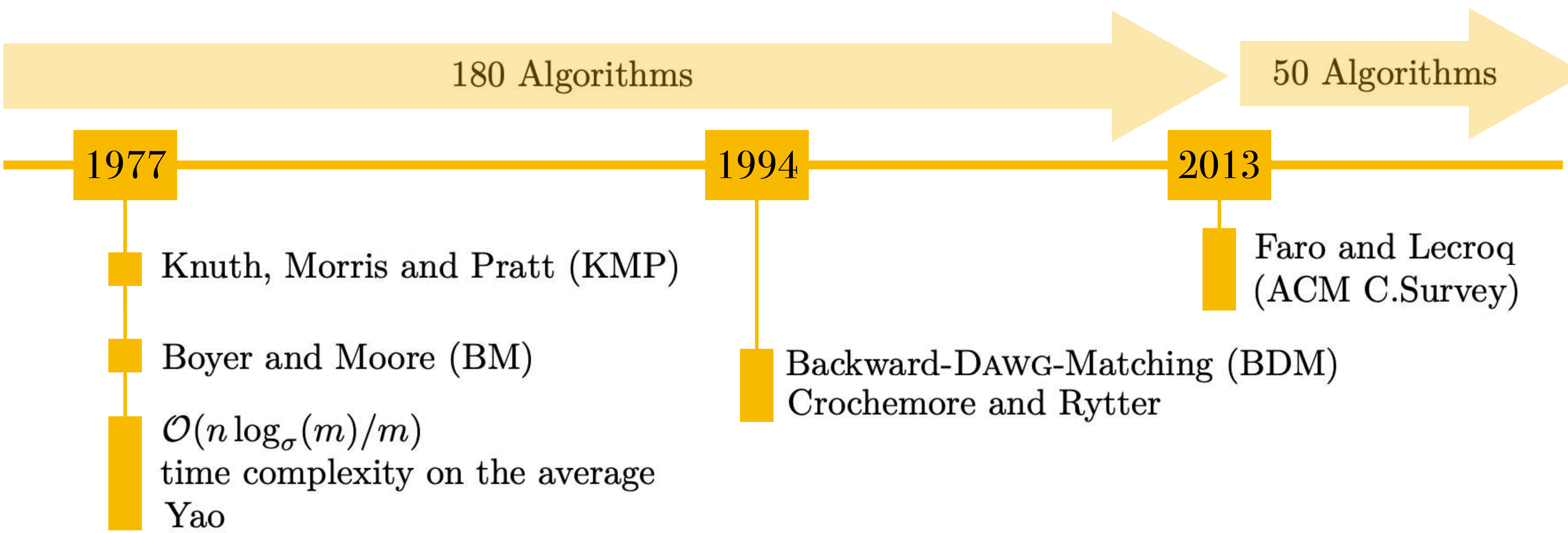
- Faro and Lecroq (ACM C.Survey)

Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.



Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

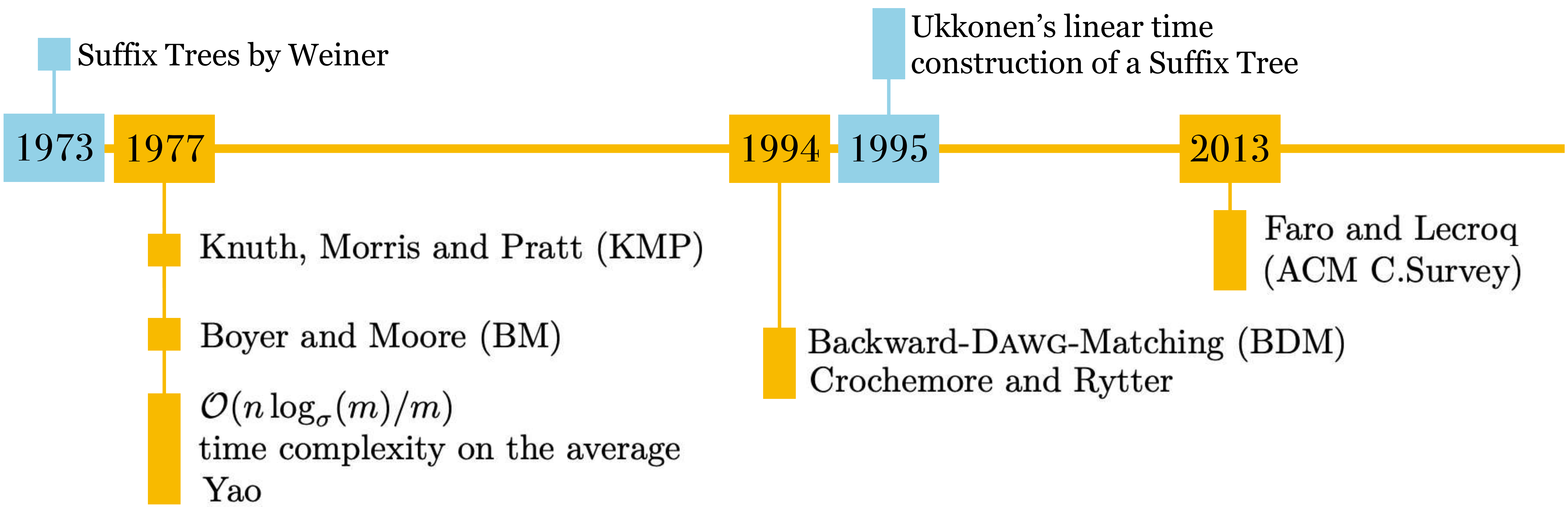
- **1973** — Suffix Trees by Weiner
- **1977** — Knuth, Morris and Pratt (KMP)
- **1977** — Boyer and Moore (BM)
- **1977** — $\mathcal{O}(n \log_\sigma(m)/m)$ time complexity on the average, Yao
- **1994** — Backward-DAWG-Matching (BDM), Crochemore and Rytter
- **2013** — Faro and Lecroq (ACM C.Survey)

Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

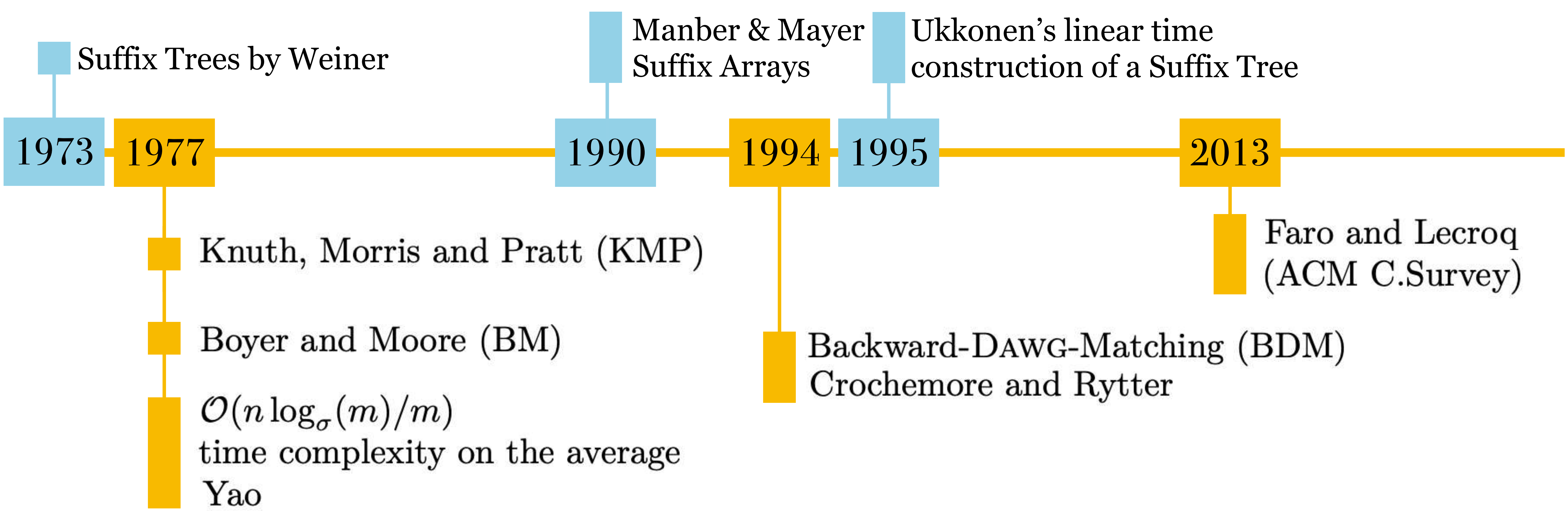


Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.



Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

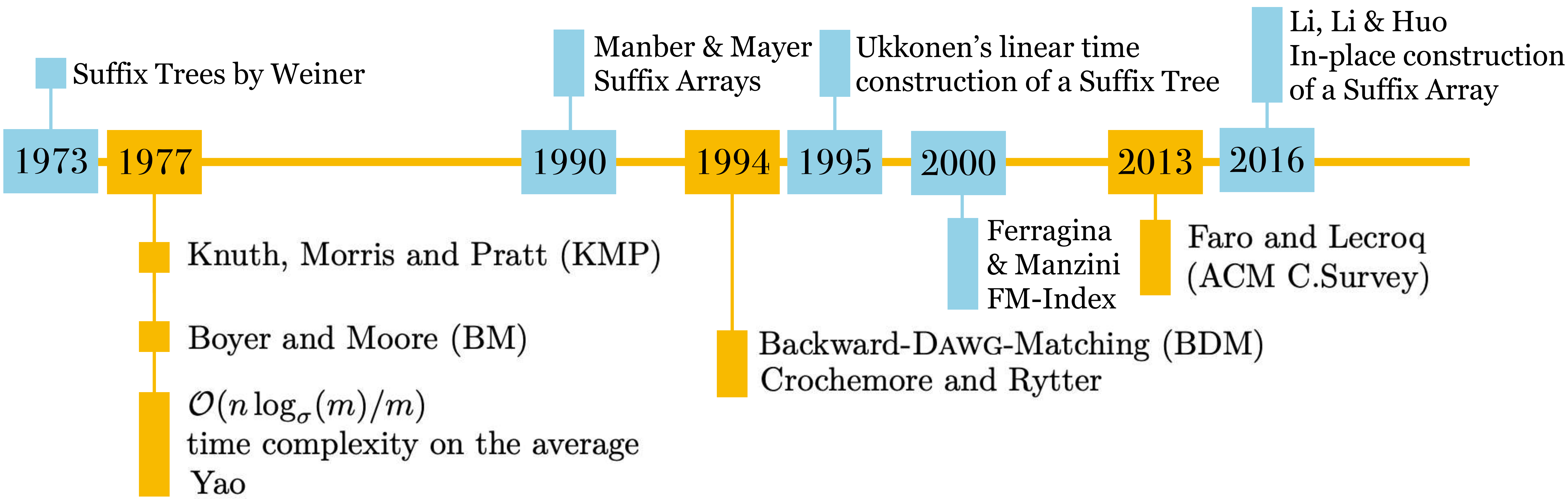
- 1973 — Suffix Trees by Weiner
- 1977 — Knuth, Morris and Pratt (KMP)
- 1977 — Boyer and Moore (BM)
- 1977 — $\mathcal{O}(n \log_\sigma(m)/m)$ time complexity on the average, Yao
- 1990 — Manber & Mayer Suffix Arrays
- 1994 — Backward-DAWG-Matching (BDM), Crochemore and Rytter
- 1995 — Ukkonen's linear time construction of a Suffix Tree
- 2013 — Faro and Lecroq (ACM C.Survey)
- 2016 — Li, Li & Huo In-place construction of a Suffix Array

Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.

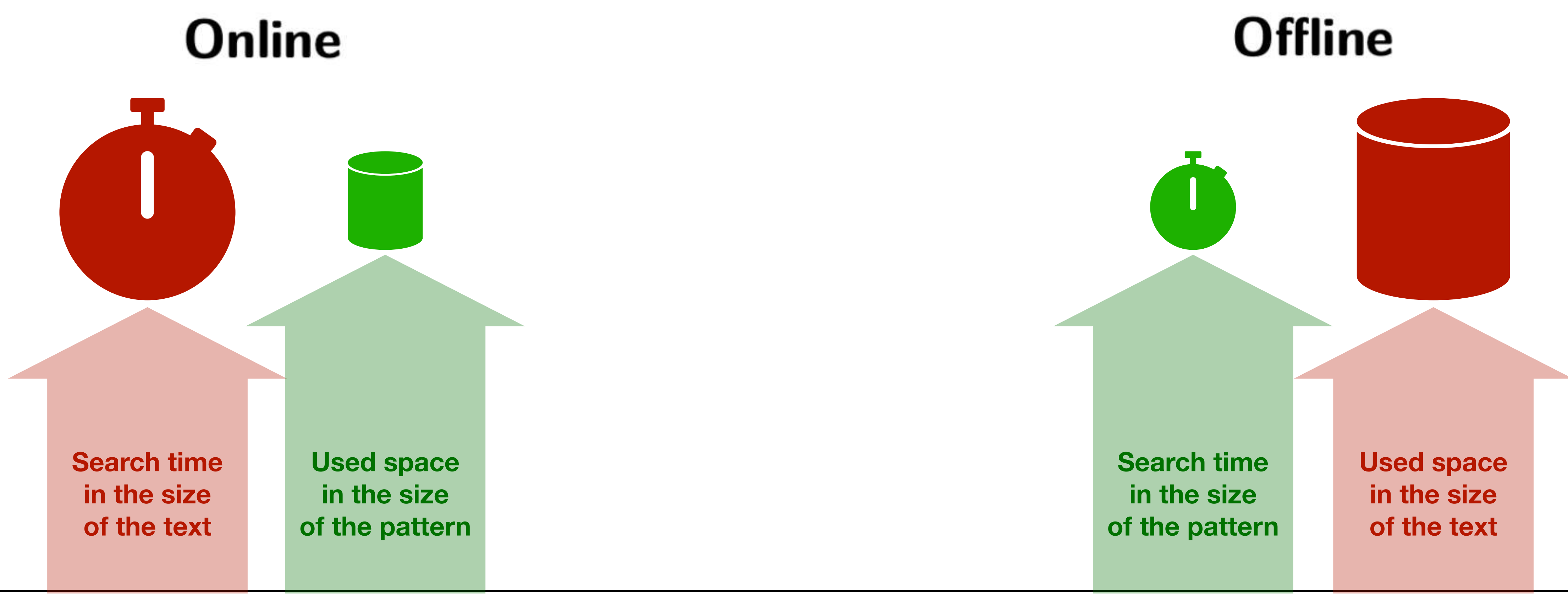


Online String Matching

The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

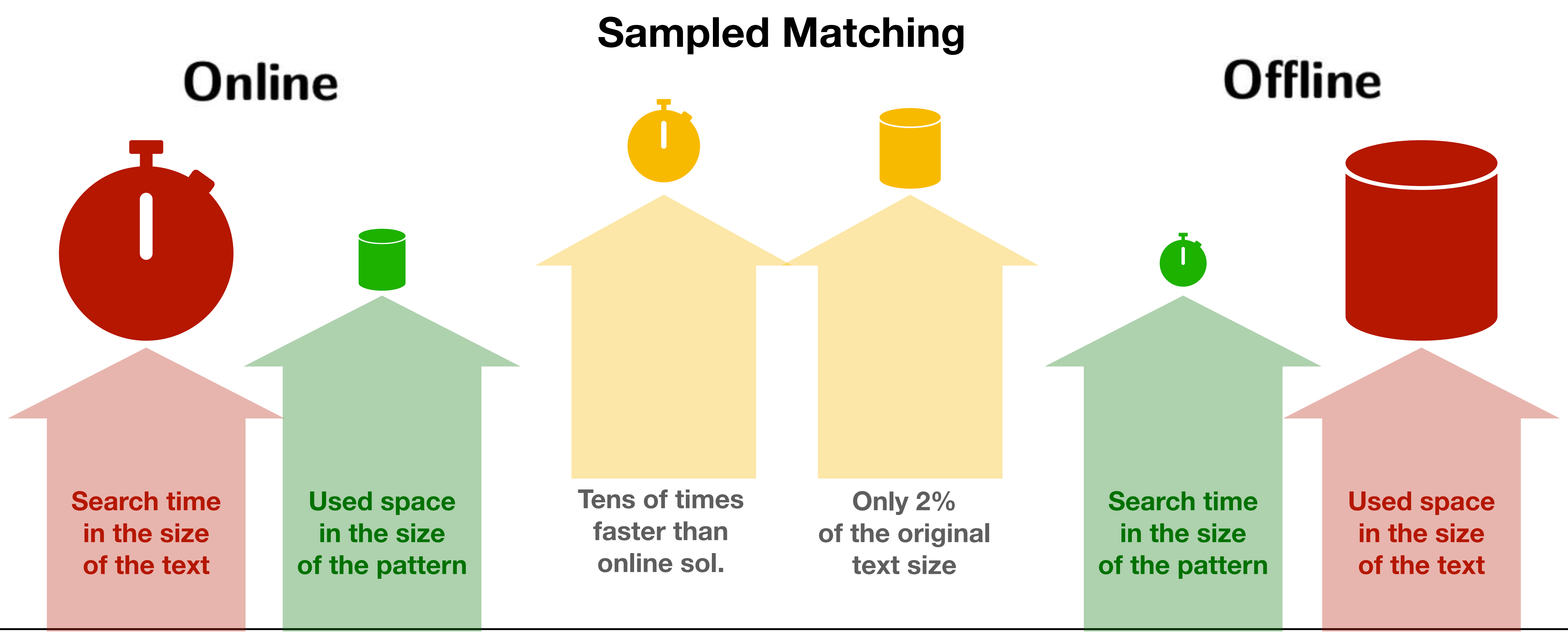
The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.



The *string matching* problem consists in finding all the occurrences of a pattern x of length m in a text y of length n, both strings defined over an alphabet Σ of size σ.

Online and Offline String Matching

The problem can be addressed in *online* mode, when you do not have access to the text before carrying out the search, or in *offline* mode, in which case it is possible to preprocess the text to speed up the search phase. In this work we consider online string matching.



Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.

Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.

The sampled-text approach possesses several characteristics:

- it typically necessitates straightforward implementation

Easy to implement

Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.

The sampled-text approach possesses several characteristics:

- it typically necessitates straightforward implementation
- it demands only a limited amount of additional space

Easy to implement

Limited space

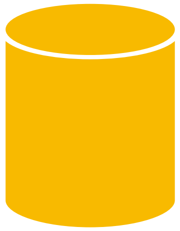
Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.

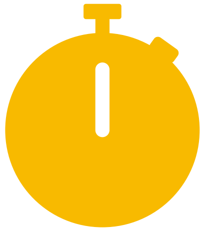
The sampled-text approach possesses several characteristics:

- it typically necessitates straightforward implementation
- it demands only a limited amount of additional space
- it enables fast search and update operations

Easy to implement



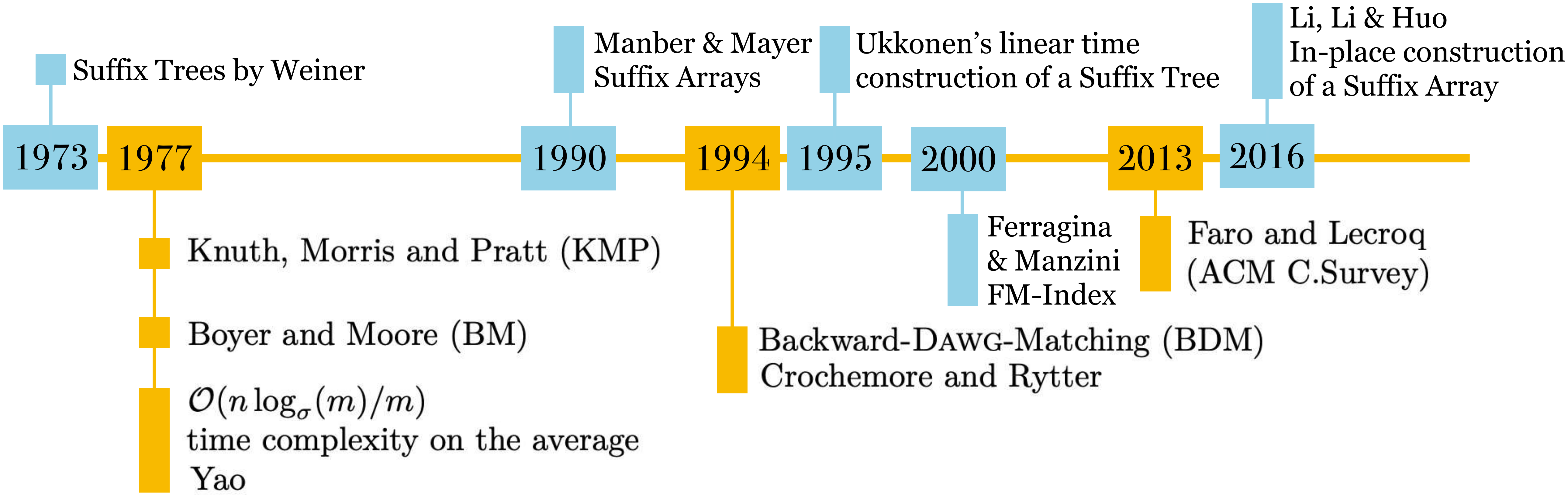
Limited space



Very fast operations

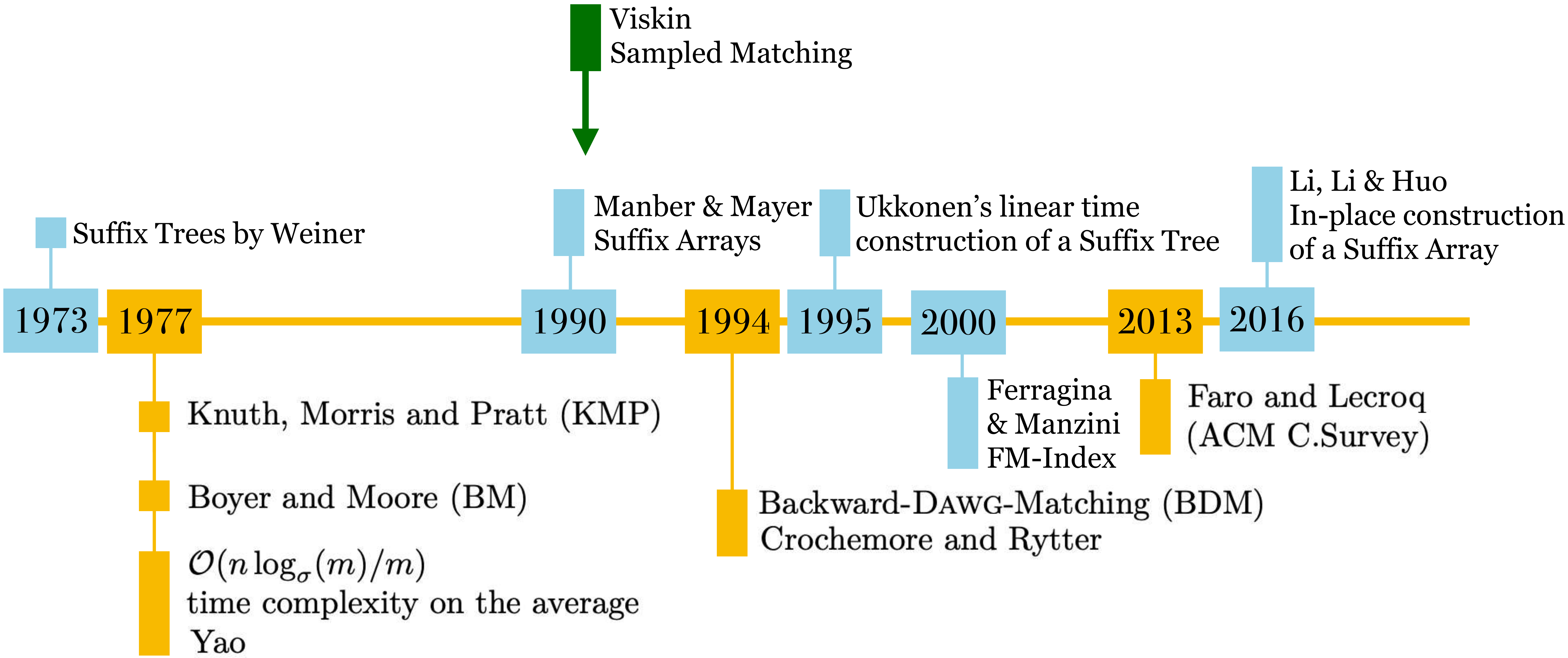
Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.



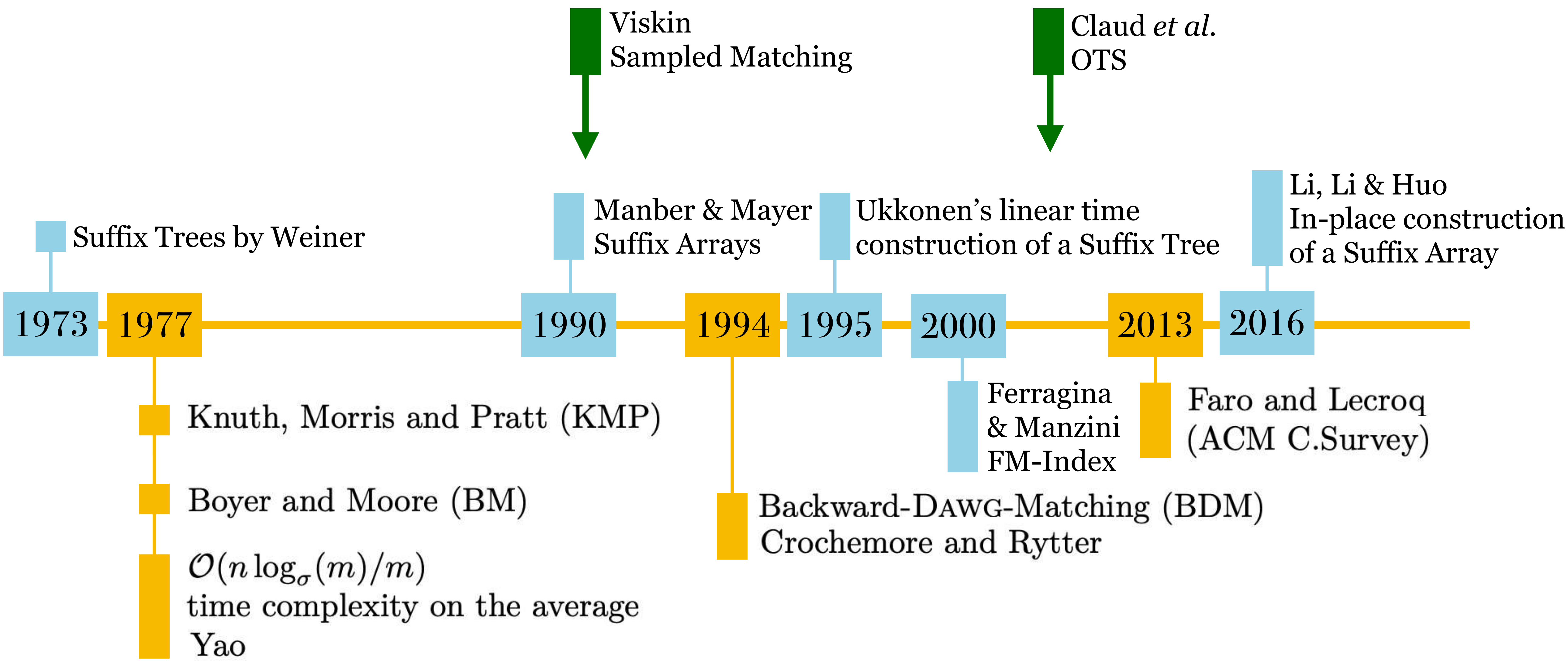
Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.



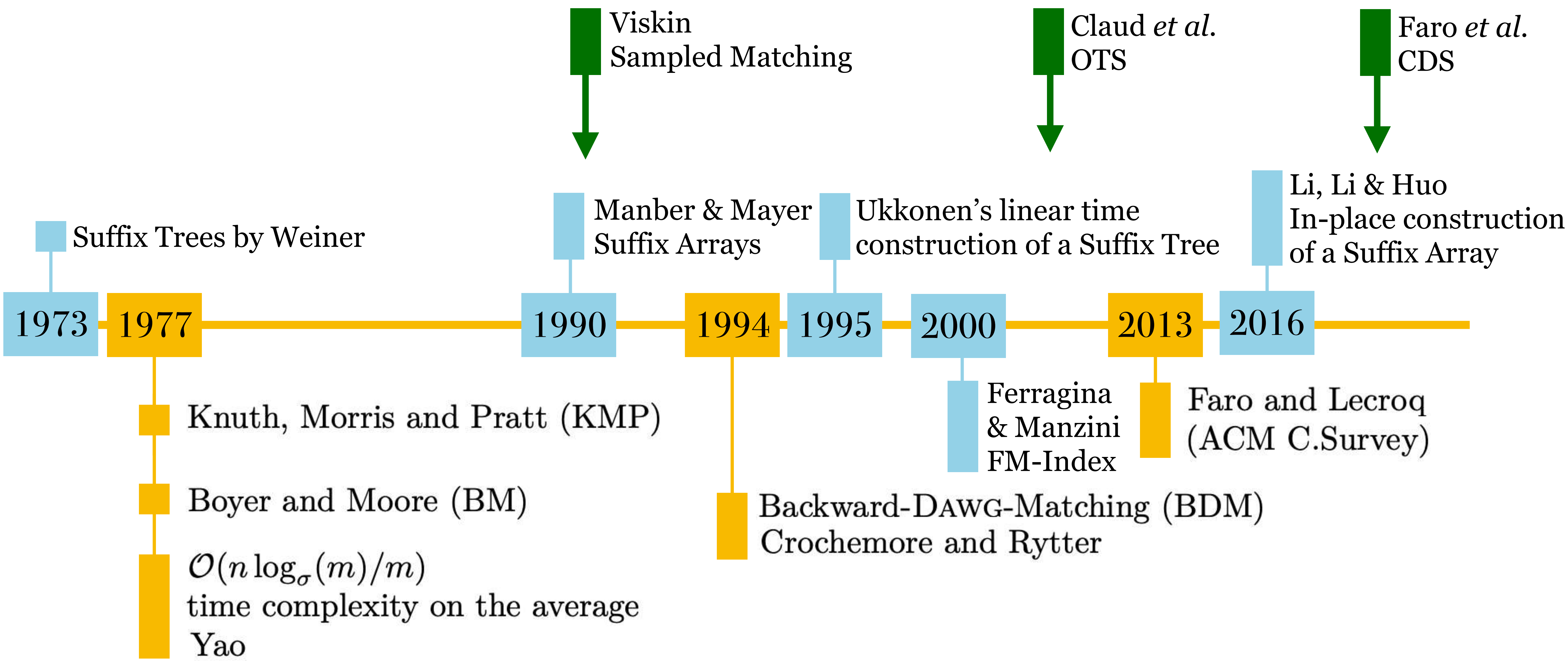
Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.



Sampled String Matching

Sampled String Matching involves creating a succinct version of the text and then applying online string matching algorithms directly to the new version. This technique enables faster discovery of pattern occurrences, but each discovery within the sampled version of the text requires subsequent verification within the original text.



OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.

y	a	b	a	a	c	a	b	d	a	a	c	a	b	c	c

$\Sigma = \{a,b,c,d\}$

OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.

y	a	b	a	a	c	a	b	d	a	a	c	a	b	c	c

$\Sigma = \{a, b, c, d\}$

$\hat{\Sigma} = \{b, c, d\}$

OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.

y															
	a	b	a	a	c	a	b	d	a	a	c	a	b	c	c

$\Sigma = \{a, b, c, d\}$

$\hat{y}$								
	b	c	b	d	c	b	c	c

$\hat{\Sigma} = \{b, c, d\}$

OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.

x	a	c	a	b											
y	a	b	a	a	c	a	b	d	a	a	c	a	b	c	c

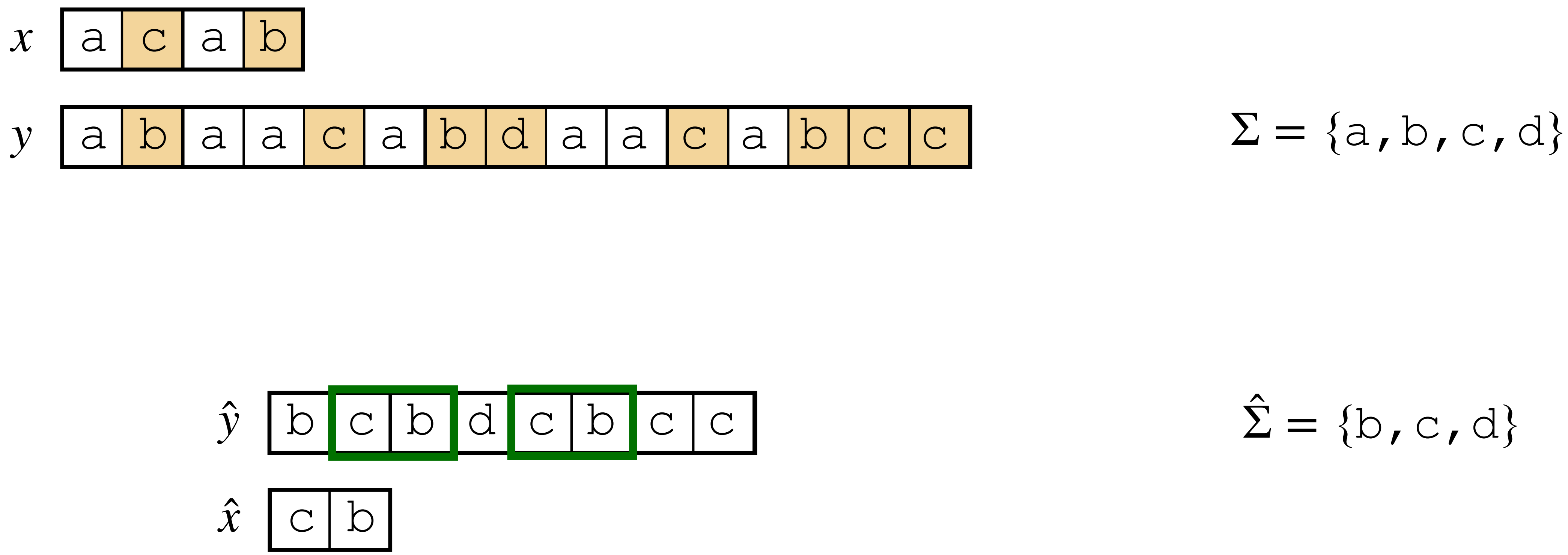
$\Sigma = \{a, b, c, d\}$

$\hat{y}$	b	c	b	d	c	b	c	c
$\hat{x}$	c	b						

$\hat{\Sigma} = \{b, c, d\}$

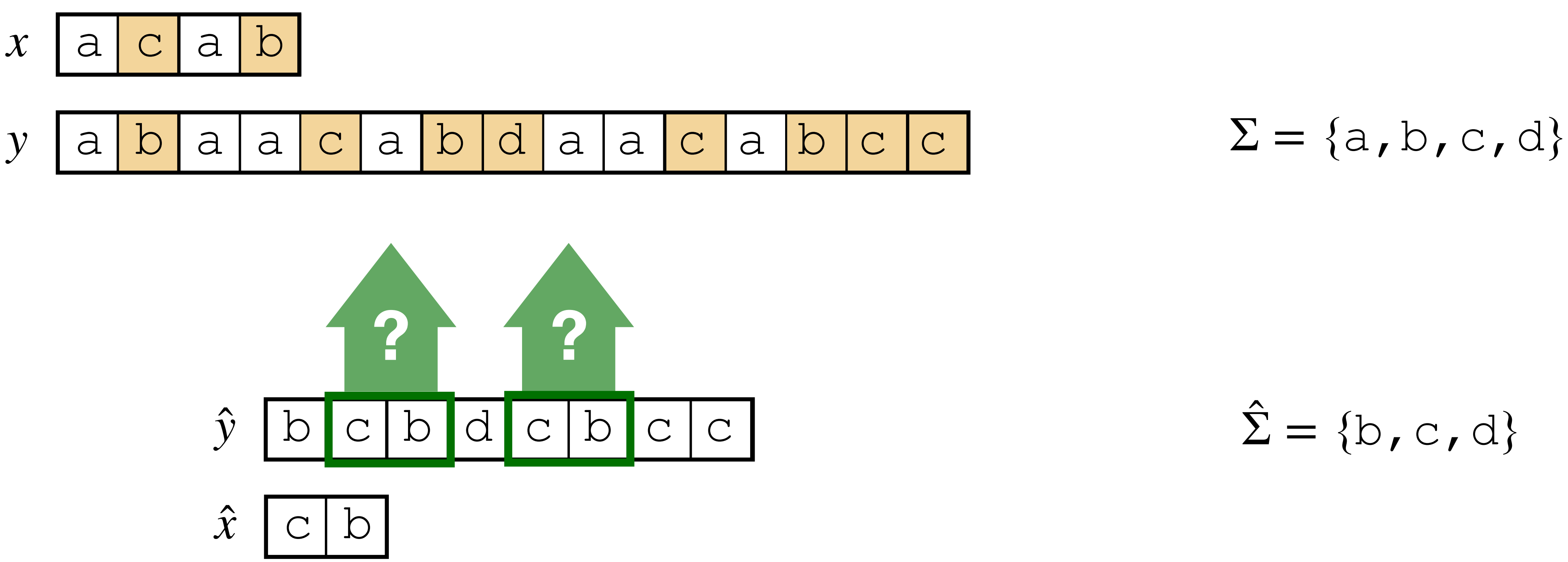
OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.



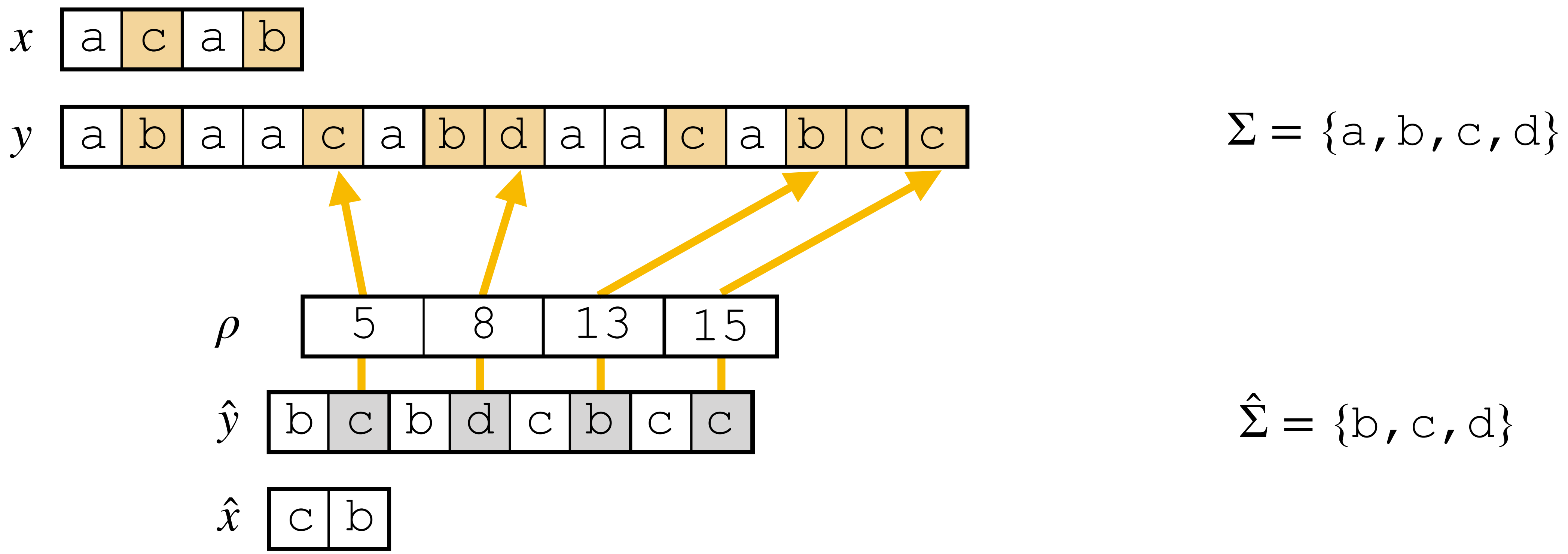
OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.



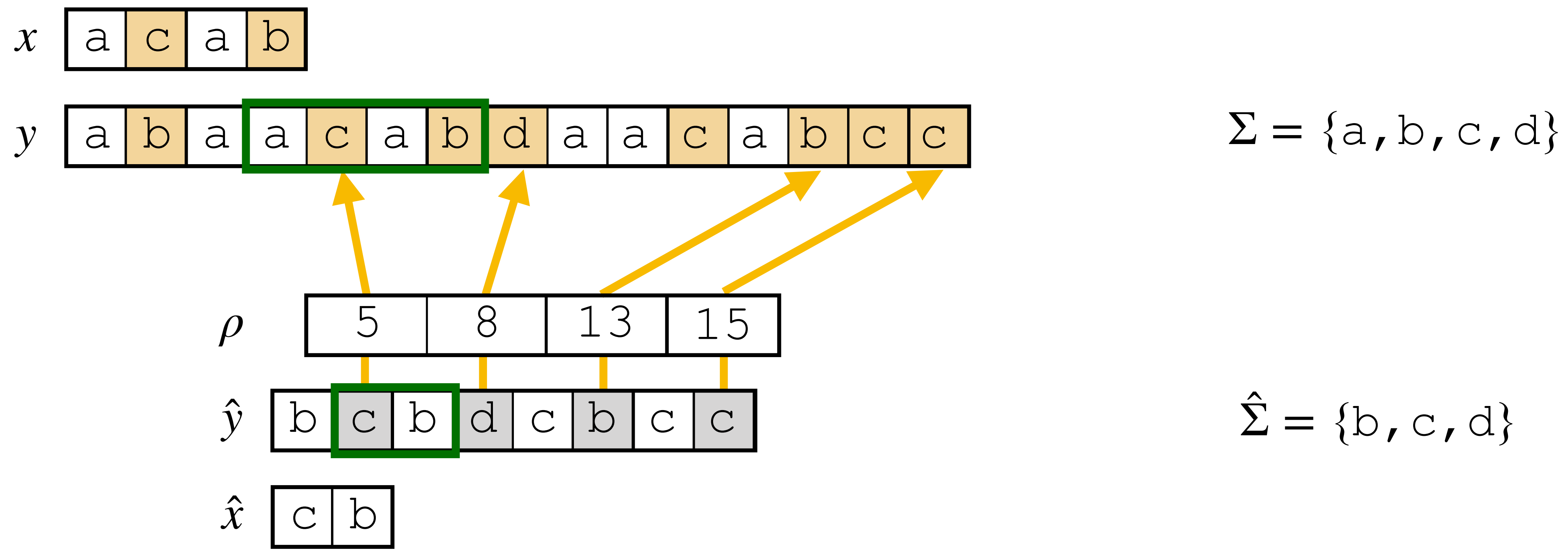
OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.



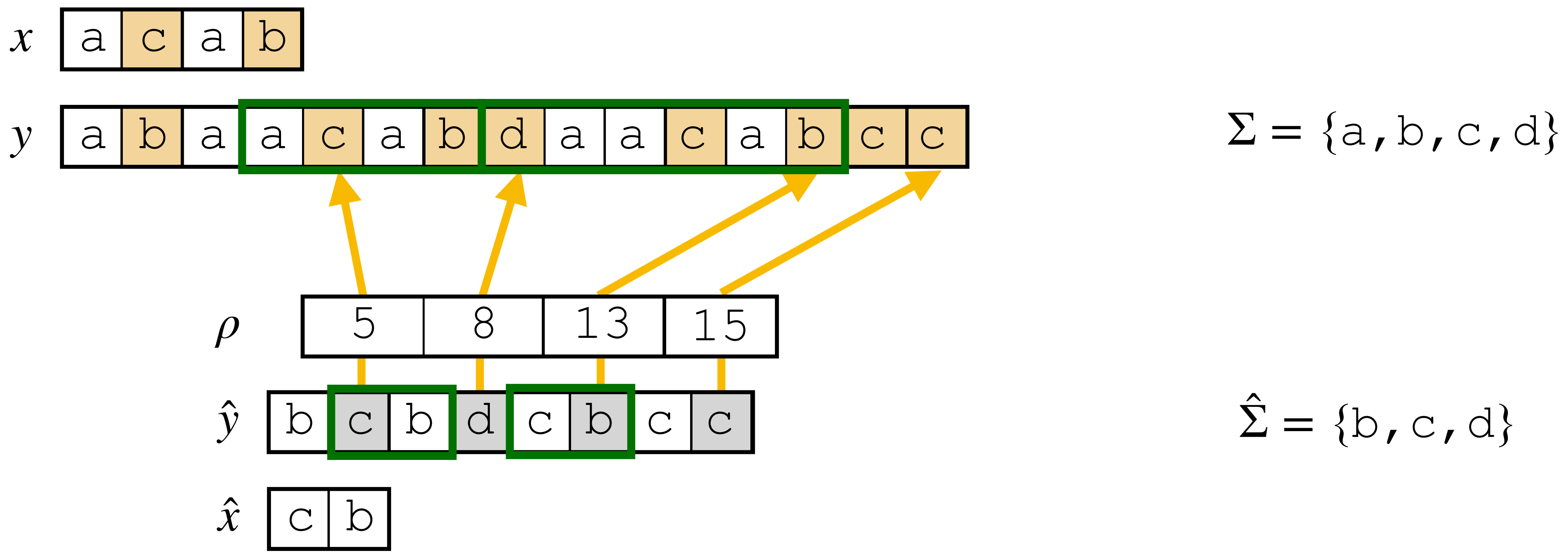
OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.

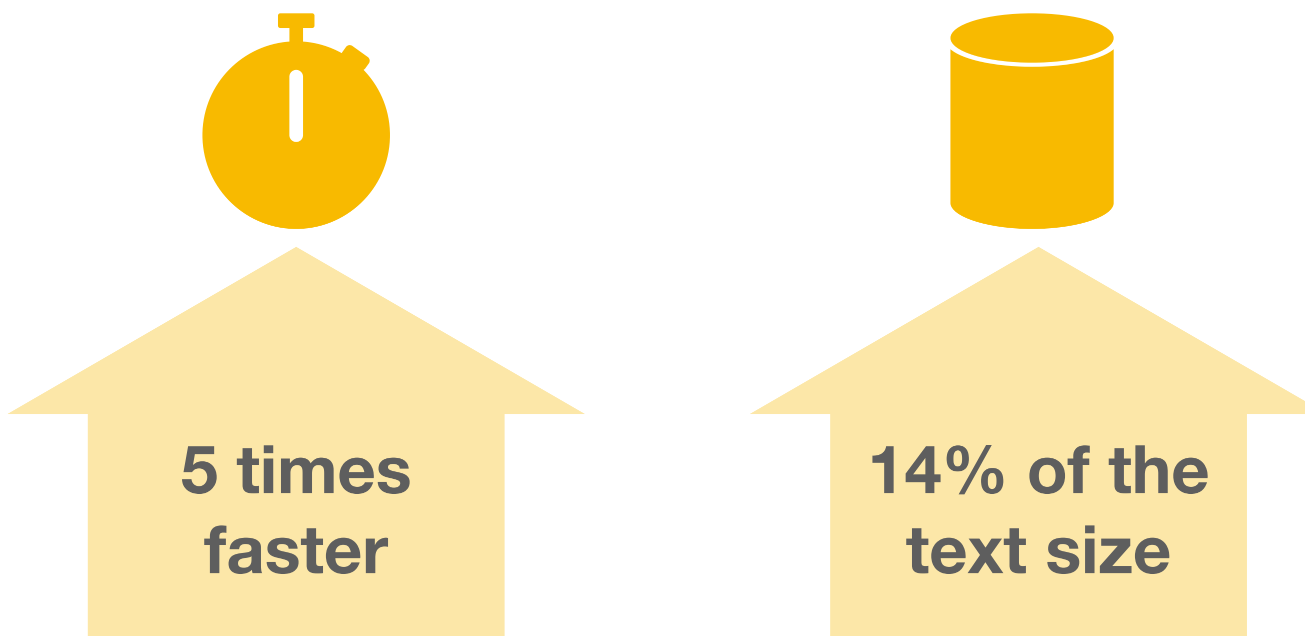


OTS: Occurrence Text-Sampling

Let y be the input text, of length n, and let x be the input pattern, of length m, both over an alphabet Σ of size σ. The main idea of their sampling approach is to select a subset of the alphabet, $\hat{\Sigma} \subset \Sigma$ (the sampled alphabet), and then to construct a partial-index as the subsequence of the text (the sampled text) $\hat{y}$, of length $\hat{n}$, containing all (and only) the characters of the sampled alphabet $\hat{\Sigma}$. More formally $\hat{y}[i] \in \hat{\Sigma}$, for all $1 \leq i \leq \hat{n}$.



OTS: Occurrence Text-Sampling



CDS: Character Distance Text-Sampling

CDS selects a sub-alphabet $C \subseteq \Sigma$ to serve as the *set of pivot characters*. Using these designated pivots, it is possible to sample the text y by calculating the distances between the n_c consecutive occurrences of any pivot character $c \in C$ within y. Formally, this sampling methodology is based on the definition of *position sampling* within a text. Given $\delta : \{1, .., n_c\} \rightarrow \{1, .., n\}$, where $\delta(i)$ is the position of the i-th occurrence of any pivot character c in y. Then the position sampled version of y, indicated by $\dot{y}$, is a numeric sequence, of length n_c, defined as $\dot{y} = \langle \delta(1), \delta(2), .., \delta(n_c) \rangle$.

y	a	g	a	a	c	g	c	a	g	t	a	t	a

$\Sigma = \{$a,c,g,t$\}$

CDS: Character Distance Text-Sampling

CDS selects a sub-alphabet $C \subseteq \Sigma$ to serve as the *set of pivot characters*. Using these designated pivots, it is possible to sample the text y by calculating the distances between the n_c consecutive occurrences of any pivot character $c \in C$ within y. Formally, this sampling methodology is based on the definition of *position sampling* within a text. Given $\delta : \{1, .., n_c\} \rightarrow \{1, .., n\}$, where $\delta(i)$ is the position of the i-th occurrence of any pivot character c in y. Then the position sampled version of y, indicated by $\dot{y}$, is a numeric sequence, of length n_c, defined as $\dot{y} = \langle \delta(1), \delta(2), .., \delta(n_c) \rangle$.

y	a	g	a	a	c	g	c	a	g	t	a	t	a

$\Sigma = \{$a,c,g,t$\}$

$C = \{$a$\}$

CDS: Character Distance Text-Sampling

CDS selects a sub-alphabet $C \subseteq \Sigma$ to serve as the *set of pivot characters*. Using these designated pivots, it is possible to sample the text y by calculating the distances between the n_c consecutive occurrences of any pivot character $c \in C$ within y. Formally, this sampling methodology is based on the definition of *position sampling* within a text. Given $\delta : \{1, .., n_c\} \rightarrow \{1, .., n\}$, where $\delta(i)$ is the position of the i-th occurrence of any pivot character c in y. Then the position sampled version of y, indicated by $\dot{y}$, is a numeric sequence, of length n_c, defined as $\dot{y} = \langle \delta(1), \delta(2), .., \delta(n_c) \rangle$.

y	a	g	a	a	c	g	c	a	g	t	a	t	a
$\dot{y}$	1		3	4				8			11		13

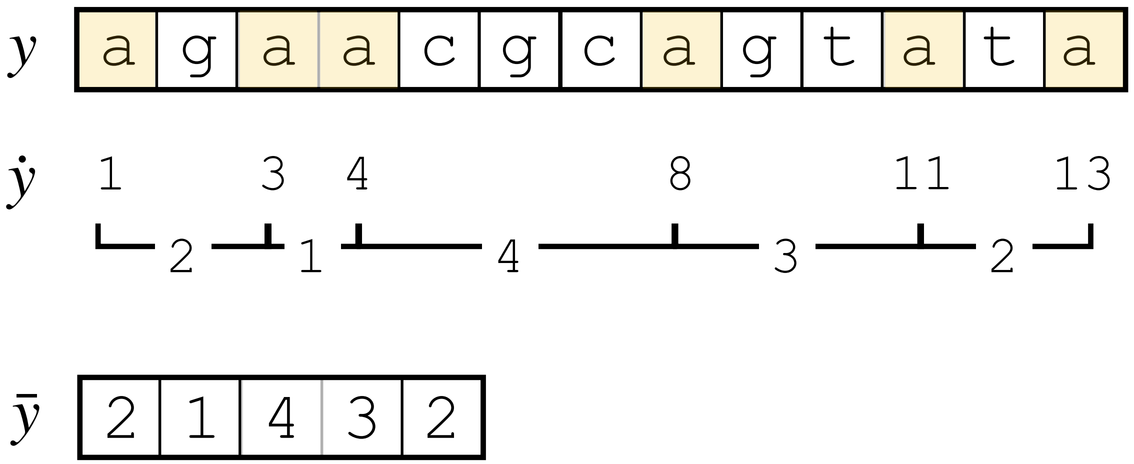
$\Sigma = \{a, c, g, t\}$

$C = \{a\}$

CDS: Character Distance Text-Sampling

CDS selects a sub-alphabet $C \subseteq \Sigma$ to serve as the *set of pivot characters*. Using these designated pivots, it is possible to sample the text y by calculating the distances between the n_c consecutive occurrences of any pivot character $c \in C$ within y. Formally, this sampling methodology is based on the definition of *position sampling* within a text. Given $\delta : \{1, .., n_c\} \rightarrow \{1, .., n\}$, where $\delta(i)$ is the position of the i-th occurrence of any pivot character c in y. Then the position sampled version of y, indicated by $\dot{y}$, is a numeric sequence, of length n_c, defined as $\dot{y} = \langle \delta(1), \delta(2), .., \delta(n_c) \rangle$.

The *characters-distance sampled version* of the text y is a numeric sequence, indicated by $\bar{y}$, of length $n_c - 1$ defined as $\bar{y} = \langle \delta(2) - \delta(1), \delta(3) - \delta(2), .., \delta(n_c) - \delta(n_c - 1) \rangle$



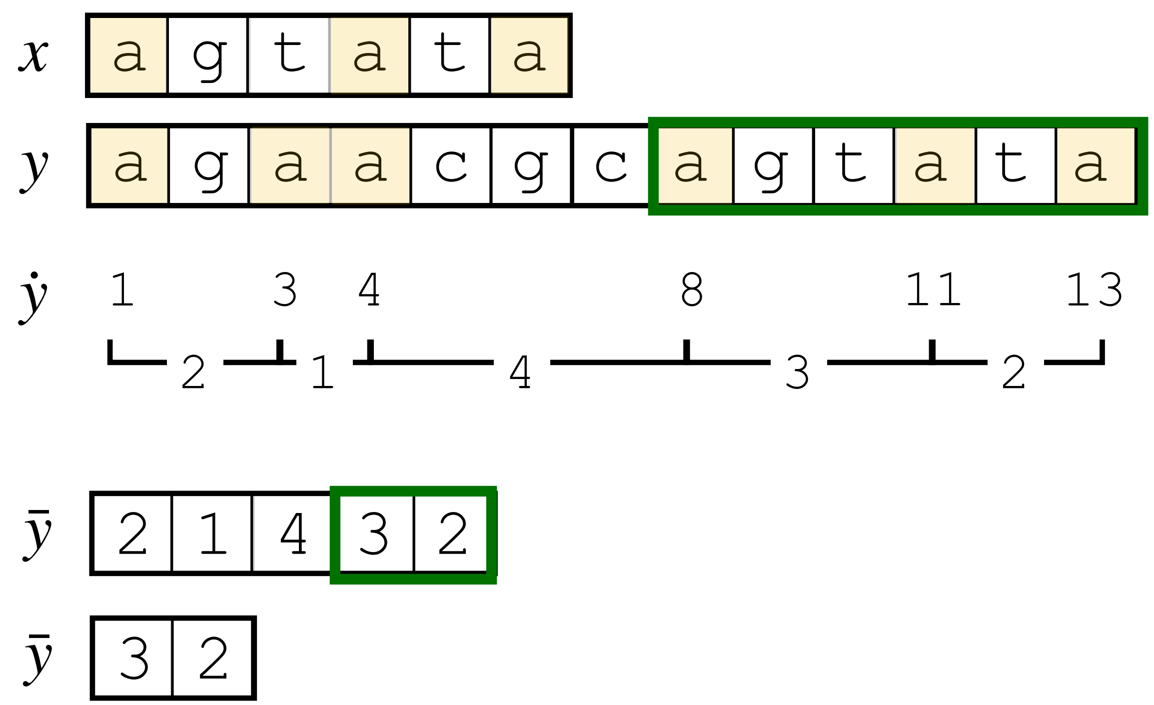
$\Sigma = \{a, c, g, t\}$

$C = \{a\}$

CDS: Character Distance Text-Sampling

CDS selects a sub-alphabet $C \subseteq \Sigma$ to serve as the *set of pivot characters*. Using these designated pivots, it is possible to sample the text y by calculating the distances between the n_c consecutive occurrences of any pivot character $c \in C$ within y. Formally, this sampling methodology is based on the definition of *position sampling* within a text. Given $\delta : \{1, .., n_c\} \rightarrow \{1, .., n\}$, where $\delta(i)$ is the position of the i-th occurrence of any pivot character c in y. Then the position sampled version of y, indicated by $\dot{y}$, is a numeric sequence, of length n_c, defined as $\dot{y} = \langle \delta(1), \delta(2), .., \delta(n_c) \rangle$.

The *characters-distance sampled version* of the text y is a numeric sequence, indicated by $\bar{y}$, of length $n_c - 1$ defined as $\bar{y} = \langle \delta(2) - \delta(1), \delta(3) - \delta(2), .., \delta(n_c) - \delta(n_c - 1) \rangle$



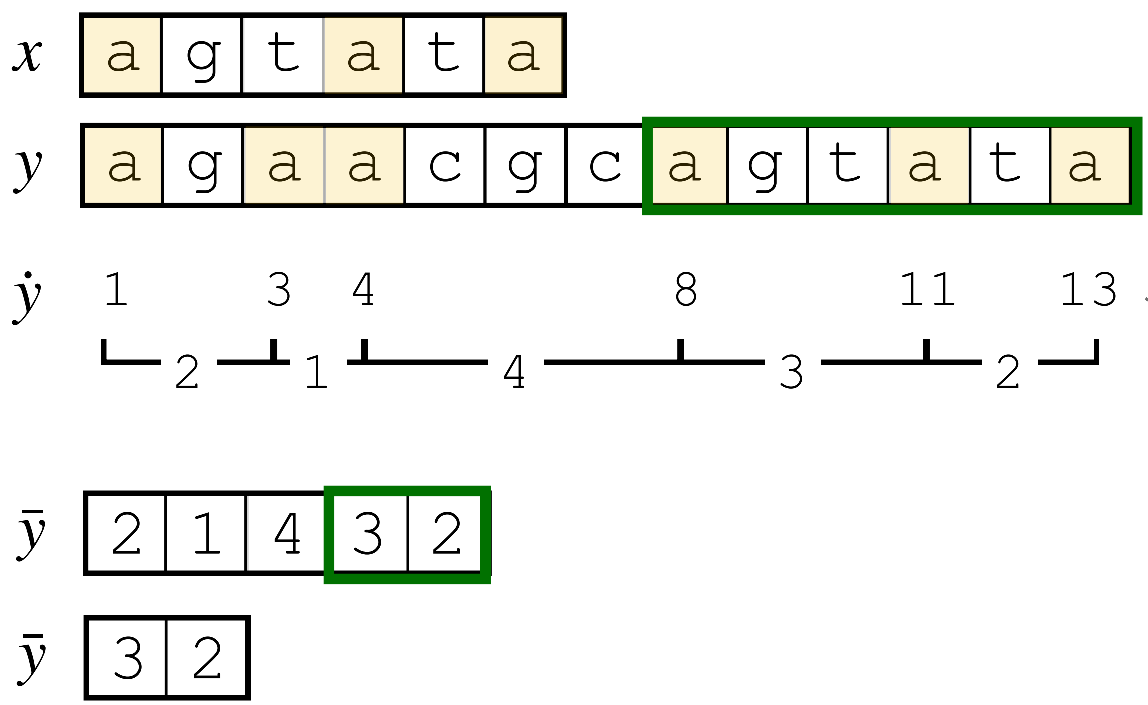
$\Sigma = \{a, c, g, t\}$

$C = \{a\}$

CDS: Character Distance Text-Sampling

CDS selects a sub-alphabet $C \subseteq \Sigma$ to serve as the *set of pivot characters*. Using these designated pivots, it is possible to sample the text y by calculating the distances between the n_c consecutive occurrences of any pivot character $c \in C$ within y. Formally, this sampling methodology is based on the definition of *position sampling* within a text. Given $\delta : \{1, .., n_c\} \to \{1, .., n\}$, where $\delta(i)$ is the position of the i-th occurrence of any pivot character c in y. Then the position sampled version of y, indicated by $\dot{y}$, is a numeric sequence, of length n_c, defined as $\dot{y} = \langle \delta(1), \delta(2), .., \delta(n_c) \rangle$.

The *characters-distance sampled version* of the text y is a numeric sequence, indicated by $\bar{y}$, of length $n_c - 1$ defined as $\bar{y} = \langle \delta(2) - \delta(1), \delta(3) - \delta(2), .., \delta(n_c) - \delta(n_c - 1) \rangle$



$\Sigma = \{\texttt{a}, \texttt{c}, \texttt{g}, \texttt{t}\}$

$C = \{\texttt{a}\}$

In order to map sampled positions to real positions CDS maintains $\dot{y}$ and compute $\bar{y}$ on the fly

CDS: Character Distance Text-Sampling

Case 0

The pivot character does not occur in x, *thus* $|\bar{x}| = 0$
We search for x in the white intervals, where the pivot character does not occur

y

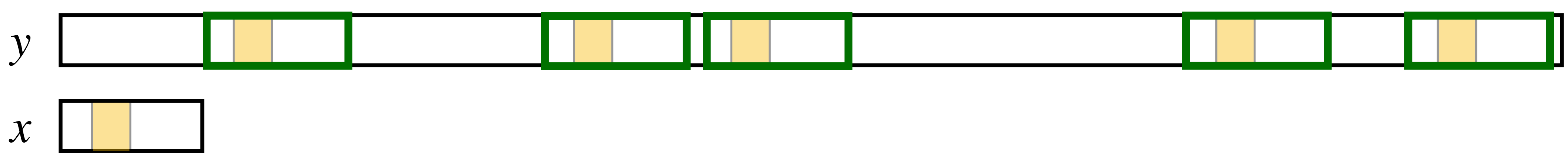
CDS: Character Distance Text-Sampling

Case 0

The pivot character does not occur in x, *thus* $|\bar{x}| = 0$
We search for x in the white intervals, where the pivot character does not occur

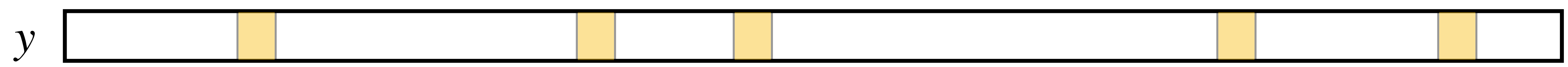
Case 1

The pivot character occurs only once in x, *thus* $|\bar{x}| = 1$
We use the sampled text as a set of anchors to locate candidate positions in y



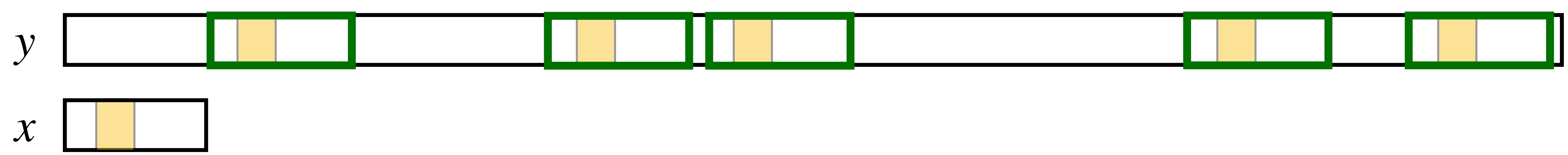
Case 0

The pivot character does not occur in x, *thus* $|\bar{x}| = 0$
We search for x in the white intervals, where the pivot character does not occur



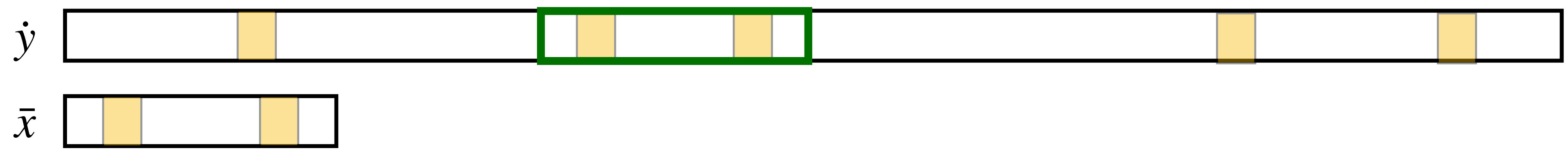
Case 1

The pivot character occurs only once in x, *thus* $|\bar{x}| = 1$
We use the sampled text as a set of anchors to locate candidate positions in y



Case 2

The pivot character occurs more than once in x, *thus* $|\bar{x}| > 1$
We search for $\bar{x}$ in $\bar{y}$ and verify any candidate occurrence in the original text



CDS: Character Distance Text-Sampling

9 times faster

2% of the text size

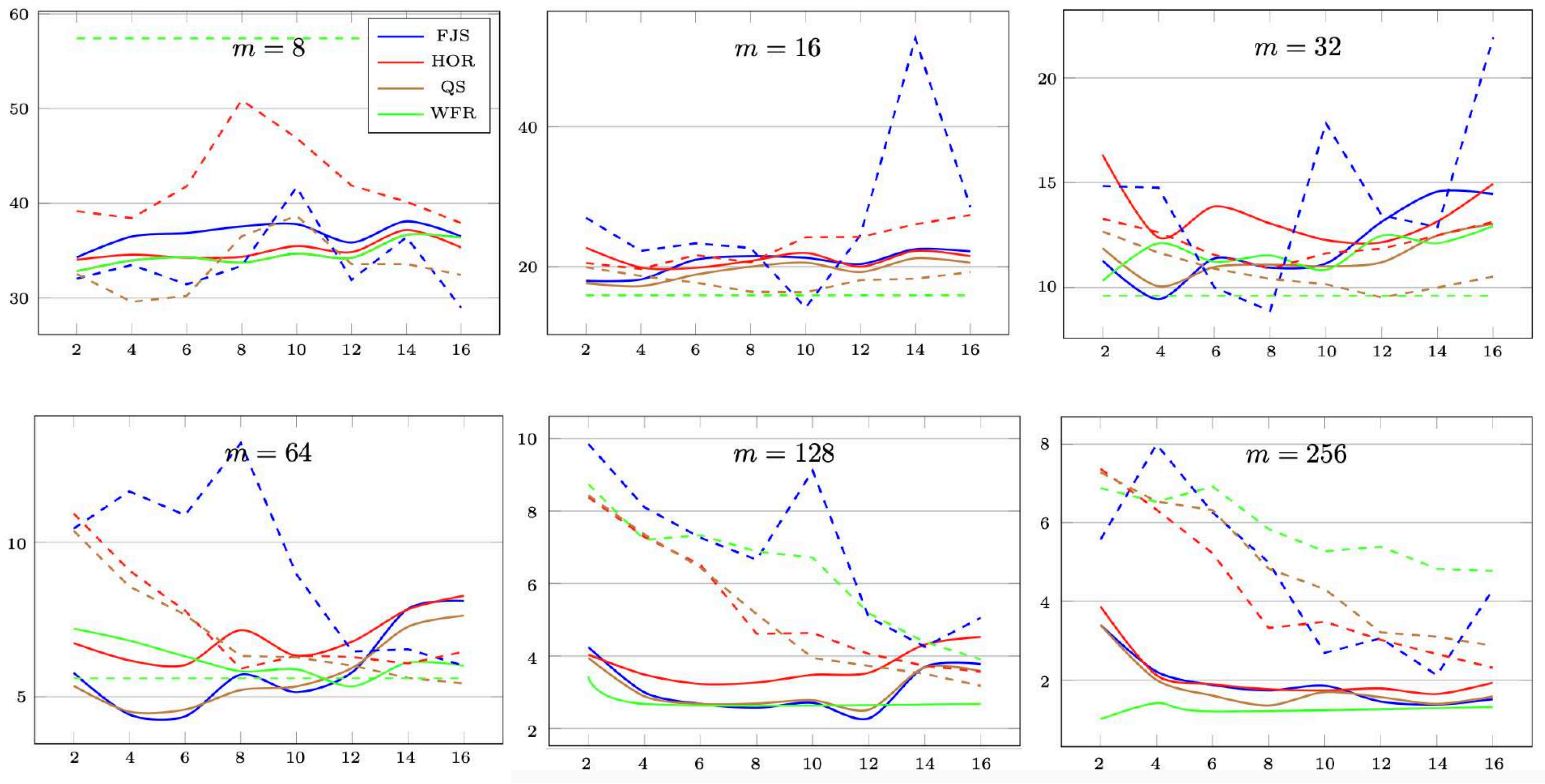
Experimental Results

Two text buffers, each with a size of 100 MB, were used, sourced from the *Pizza and Chili* dataset [15], available online for download. Specifically, the algorithms were tested using a genomics data sequence and a natural language text. For each sequence, 500 patterns were randomly selected from the text, and the average running time was computed over the 500 runs.

We tested the OTS and CDS approach using the following algorithms:

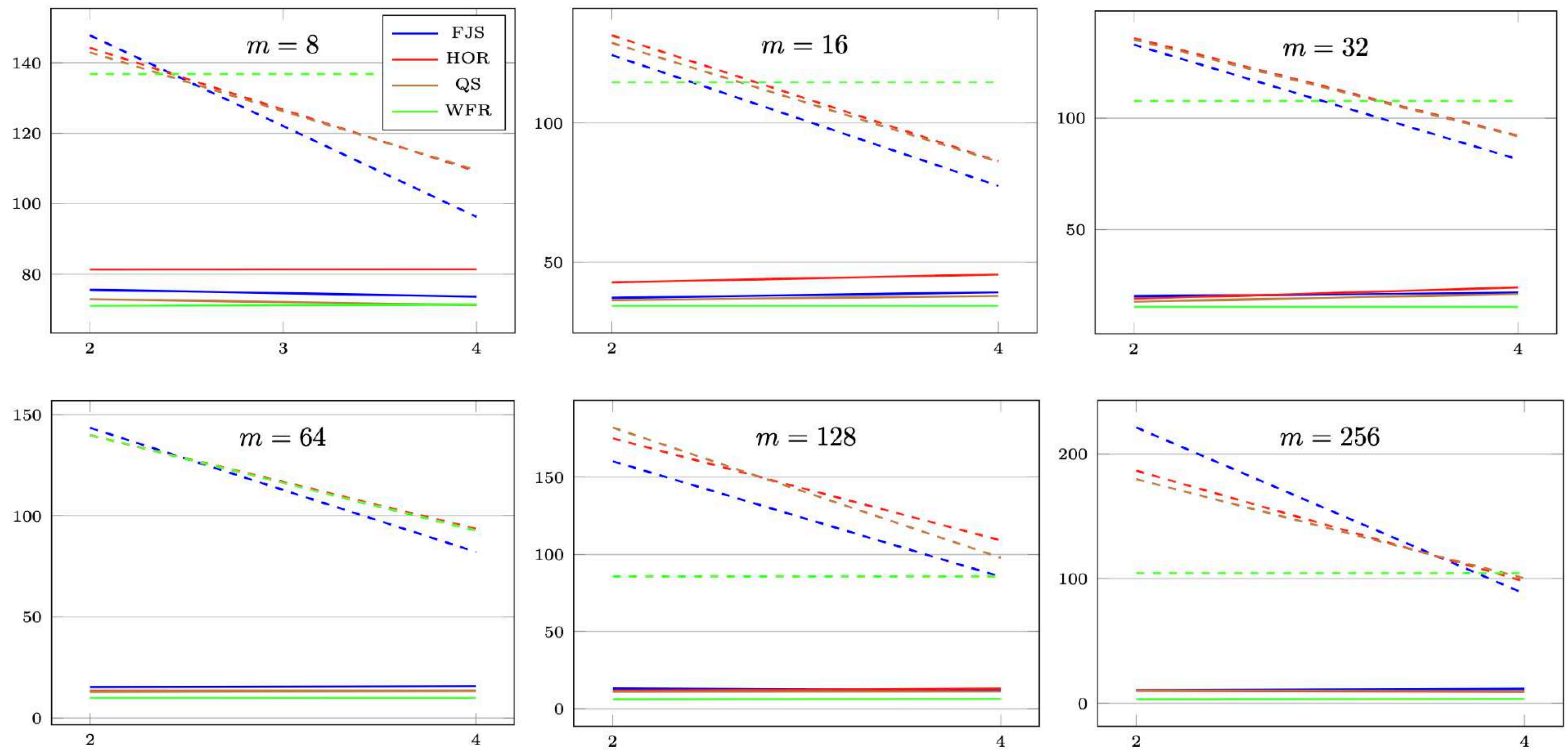
- `HOR` Horspool
- `QS` Quick Search
- `FJS` Franek-Jennings-Smyth
- `WFR` Weak-Factor Recognition

Online Searching on Natural Language Sequence



Experimental Results

Online Searching on Genomics Sequence



Conclusions

In this paper we have presented an extension of the text sampling approaches, called Character Distance and Occurrence Text Sampling, to the case of applying different searching algorithms. This extension was carried out using four different well known algorithms. Our results proved the efficacy of the sampled methods discussed in this paper and their versatility to be adapted to any online string matching algorithms without impacting their original performances.

Although our tests were limited to the exact string matching problem, obtaining excellent results, we believe that the approach can be effectively generalized even to non-standard string matching. Our future studies will focus in this direction in order to apply sampled string matching to other problems related to text processing.

Thanks