Improved and Self-Tuned Occurrence Heuristics

Simone Faro and Domenico Cantone

Department of Mathematics and Computer Science
Department of Linguistics and Humanities
University of Catania (Italy)

web-page: `http://www.dmi.unict.it/~faro`
email: `faro@dmi.unict.it`

Prague Stringology Conference
September 2, 2013

The Online Exact String Matching Problem

Definition

Given a text t of length n and a pattern p of length m over some alphabet Σ of size σ, the exact string matching problem consists in finding all occurrences of the pattern p in t.

It has been extensively studied in computer science because of its direct application to many areas.

It is basic components in many software applications

It plays an important role in theoretical computer science by providing challenging problems.

The Online Exact String Matching Problem

Definition

Given a text t of length n and a pattern p of length m over some alphabet Σ of size σ, the exact string matching problem consists in finding all occurrences of the pattern p in t.

An example

a	b	c	a	c	a	b	b	a	c	a	b	b	a	c	a	b	c	a	a	b	a

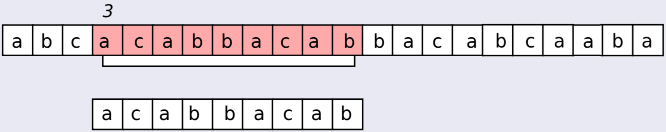
a	c	a	b	b	a	c	a	b

The Online Exact String Matching Problem

Definition

Given a text t of length n and a pattern p of length m over some alphabet Σ of size σ, the exact string matching problem consists in finding all occurrences of the pattern p in t.

An example

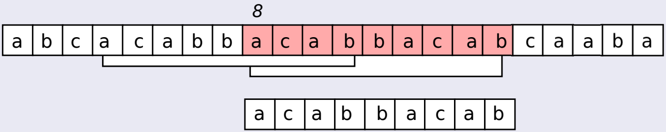


The Online Exact String Matching Problem

Definition

Given a text t of length n and a pattern p of length m over some alphabet Σ of size σ, the exact string matching problem consists in finding all occurrences of the pattern p in t.

An example



The Boyer-Moore Algorithm

The Boyer-Moore algorithm checks whether s is a valid shift, by scanning the pattern P from right to left and, at the end of the matching phase, it computes the shift increment as the maximum value suggested by the *good-suffix heuristic* and the *bad-character heuristic* below, using the functions gs_P and bc_P respectively.

The Boyer-Moore Algorithm

The Boyer-Moore algorithm checks whether s is a valid shift, by scanning the pattern P from right to left and, at the end of the matching phase, it computes the shift increment as the maximum value suggested by the *good-suffix heuristic* and the *bad-character heuristic* below, using the functions gs_P and bc_P respectively.

An example

s

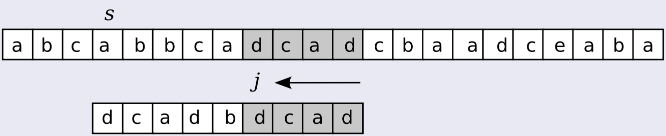
a	b	c	a	b	b	c	a	d	c	a	d	c	b	a	a	d	c	e	a	b	a

d	c	a	d	b	d	c	a	d

The Boyer-Moore Algorithm

The Boyer-Moore algorithm checks whether s is a valid shift, by scanning the pattern P from right to left and, at the end of the matching phase, it computes the shift increment as the maximum value suggested by the *good-suffix heuristic* and the *bad-character heuristic* below, using the functions gs_P and bc_P respectively.

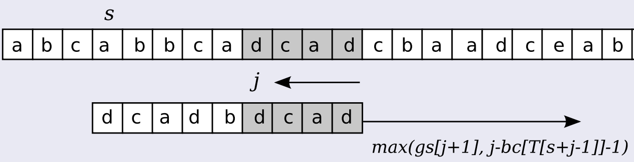
An example



The Boyer-Moore Algorithm

The Boyer-Moore algorithm checks whether s is a valid shift, by scanning the pattern P from right to left and, at the end of the matching phase, it computes the shift increment as the maximum value suggested by the *good-suffix heuristic* and the *bad-character heuristic* below, using the functions gs_P and bc_P respectively.

An example

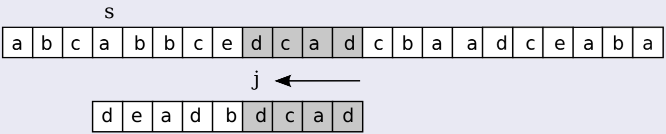


The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s + j - 1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example

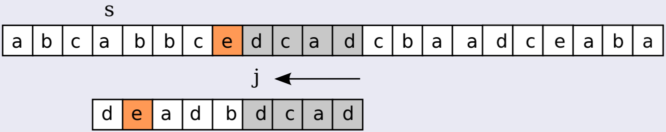


The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example



The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \le k < m \mid P[k] = c\} \cup \{-1\}),$$

An example

			s																		
a	b	c	a	b	b	c	e	d	c	a	d	c	b	a	a	d	c	e	a	b	a

$\longrightarrow$

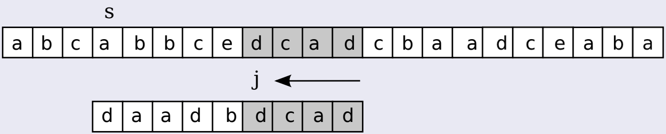
d	e	a	d	b	d	c	a	d

The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example

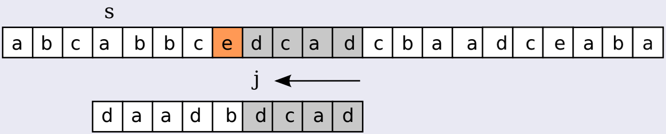


The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example



The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example

s

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

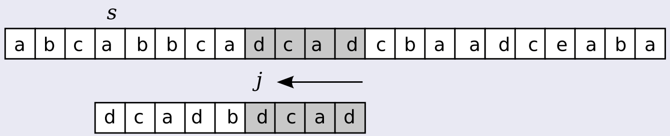
d	a	a	d	b	d	c	a	d

The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example

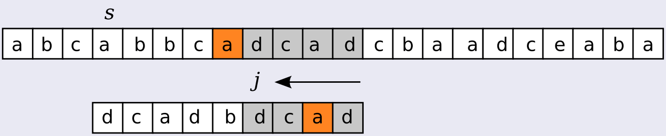


The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s+j-1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example

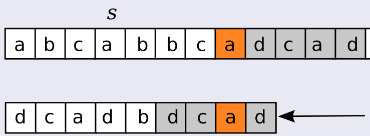


The Bad-Character Rule

The shift increment suggested by the bad-character heuristic is given by the expression $(j - bc_P(T[s + j - 1]) - 1)$, where

$$bc_P(c) = \max(\{0 \leq k < m \mid P[k] = c\} \cup \{-1\}),$$

An example



The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

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

b	c	a	d	b	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

a	b	c	a	b	b	c	a	**b**	c	a	d	c	b	a	a	d	c	e	a	b	a

b	c	a	d	**b**	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

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

b	c	a	d	b	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

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

b	c	a	d	b	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

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

b	c	a	d	b	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

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

b	c	a	d	b	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

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

b	c	a	d	b	d	c	a	d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example

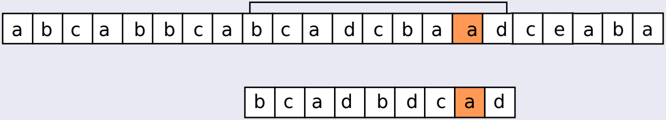
a b c a b b c a b c a d c b a a d c e a b a

b c a d b d c a d

The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

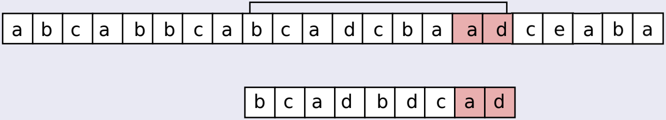
An example



The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

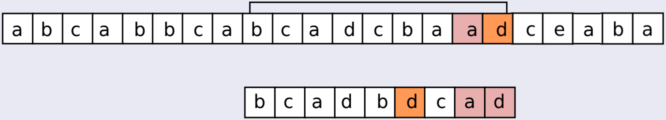
An example



The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

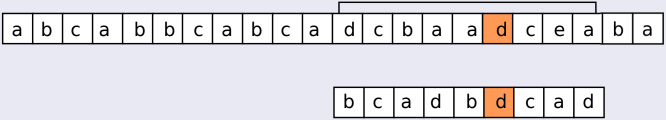
An example



The Horpool Algorithm

The Horspool algorithm uses a similar shifting strategy as Boyer-Moore. It simply drops the good-suffix rule and uses only the bad-character rule for shifting. In order to avoid negative advancement it always uses the rightmost character of the current window for computing the shift amount.

An example



Efficient Occurrence Heuristics

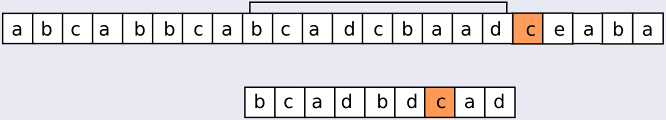
Due to the simplicity and ease of implementation of the bad-character heuristic, some variants of the Boyer-Moore algorithm have focused just around it and dropped the good-suffix heuristic.
This is the case, for instance, of the following algorithms

Horspool	1980
Zhu-Takaoka	1987
Quick-Search	1990
Tuned-Boyer-Moore	1991
Smith	1991
Berry-Ravindran	1999

The Quick-Search Algorithm

The Quick-Search algorithm, presented by Sunday in 1990, also uses a modification of the original occurrence heuristic. When a mismatching character is encountered, the pattern is always shifted to the right by at least one character, but never by more than m characters. Thus, the character $t[s+m]$ is always involved in testing for the next alignment and can be used for computing the shift.

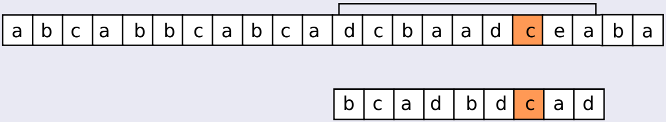
An example



The Quick-Search Algorithm

The Quick-Search algorithm, presented by Sunday in 1990, also uses a modification of the original occurrence heuristic. When a mismatching character is encountered, the pattern is always shifted to the right by at least one character, but never by more than m characters. Thus, the character $t[s+m]$ is always involved in testing for the next alignment and can be used for computing the shift.

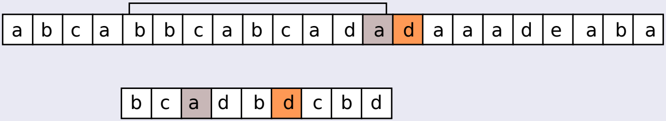
An example



The Smith Algorithm

The Smith algorithm computes its shift advancements by taking the largest value suggested by the Horpool and the Quick-Search bad-character rules.

An example



The Smith Algorithm

The Smith algorithm computes its shift advancements by taking the largest value suggested by the Horpool and the Quick-Search bad-character rules.

An example

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

b	c	a	d	b	d	c	b	d

The Zhu-Takaoka Algorithm

The Zhu-Takaoka algorithm extends the Horspool algorithm by using the last two characters $t[s+m-2]$ and $t[s+m-1]$ in place of only $t[s+m-1]$.

An example

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

b	c	a	d	b	d	b	a	d

The Zhu-Takaoka Algorithm

The Zhu-Takaoka algorithm extends the Horspool algorithm by using the last two characters $t[s+m-2]$ and $t[s+m-1]$ in place of only $t[s+m-1]$.

An example

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

b	c	a	d	b	d	b	a	d

The Berry-Ravindran Algorithm

A more effective algorithm, due to Berry and Ravindran, extends the Quick-Search algorithm in a similar manner, by using the characters $t[s+m]$ and $t[s+m+1]$ in place of only $t[s+m]$.

An example

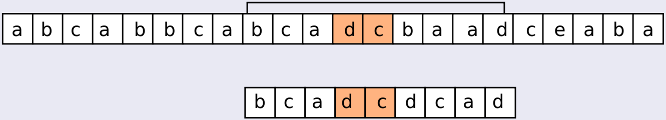
a	b	c	a	b	b	c	a	b	c	a	d	c	b	a	a	d	c	e	a	b	a

b	c	a	d	c	d	c	a	d

The Berry-Ravindran Algorithm

A more effective algorithm, due to Berry and Ravindran, extends the Quick-Search algorithm in a similar manner, by using the characters $t[s+m]$ and $t[s+m+1]$ in place of only $t[s+m]$.

An example



A Simple Improved Occurrence Heuristic

The generalized Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq 2m - 1$, $gbc_p(i, t[s+i])$ is the shift advancement such that the character $t[s+i]$ is aligned with its rightmost occurrence in $p[0 \,..\, \min(i, m) - 1]$, if present; otherwise $gbc_p(i, t[s+i])$ evaluates to $i + 1$.

$$gbc_p(i, c) =_{\text{Def}} \min(\{i - k \mid 0 \leq k < \min(i, m) \text{ and } p[k] = c\} \cup \{i + 1\}),$$

A Simple Improved Occurrence Heuristic

The generalized Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq 2m-1$, $gbc_p(i, t[s+i])$ is the shift advancement such that the character $t[s+i]$ is aligned with its rightmost occurrence in $p[0\,..\,\min(i,m)-1]$, if present; otherwise $gbc_p(i, t[s+i])$ evaluates to $i+1$.

$$gbc_p(i,c) =_{\text{Def}} \min(\{i-k \mid 0 \leq k < \min(i,m) \text{ and } p[k]=c\} \cup \{i+1\}),$$

An example

			s				$s+i$														
a	b	c	a	b	b	c	a	d	c	a	d	c	b	a	a	d	c	e	a	b	a

				i				
d	c	a	d	b	d	c	a	d

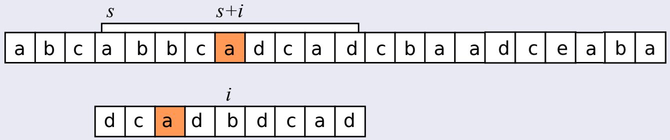
A Simple Improved Occurrence Heuristic

The generalized Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq 2m-1$, $gbc_p(i, t[s+i])$ is the shift advancement such that the character $t[s+i]$ is aligned with its rightmost occurrence in $p[0 \,..\, \min(i,m)-1]$, if present; otherwise $gbc_p(i, t[s+i])$ evaluates to $i+1$.

$$gbc_p(i,c) =_{\text{Def}} \min(\{i-k \mid 0 \leq k < \min(i,m) \text{ and } p[k]=c\} \cup \{i+1\}),$$

An example



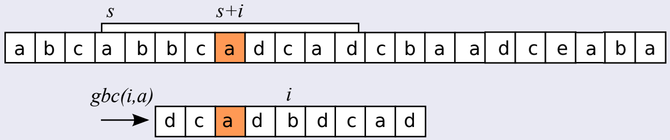
A Simple Improved Occurrence Heuristic

The generalized Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq 2m-1$, $gbc_p(i, t[s+i])$ is the shift advancement such that the character $t[s+i]$ is aligned with its rightmost occurrence in $p[0 \,..\, \min(i,m)-1]$, if present; otherwise $gbc_p(i, t[s+i])$ evaluates to $i+1$.

$$gbc_p(i,c) =_{\text{Def}} \min(\{i-k \mid 0 \leq k < \min(i,m) \text{ and } p[k]=c\} \cup \{i+1\}),$$

An example



A Simple Improved Occurrence Heuristic

The generalized Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq 2m-1$, $gbc_p(i, t[s+i])$ is the shift advancement such that the character $t[s+i]$ is aligned with its rightmost occurrence in $p[0\,..\,\min(i,m)-1]$, if present; otherwise $gbc_p(i, t[s+i])$ evaluates to $i+1$.

$$gbc_p(i,c) =_{\text{Def}} \min(\{i-k \mid 0 \leq k < \min(i,m) \text{ and } p[k]=c\} \cup \{i+1\}),$$

An example

s $\quad$ $s+i$

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

i

d	c	a	d	b	d	c	a	d

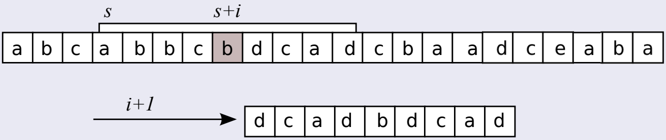
A Simple Improved Occurrence Heuristic

The generalized Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq 2m-1$, $gbc_p(i, t[s+i])$ is the shift advancement such that the character $t[s+i]$ is aligned with its rightmost occurrence in $p[0 \,..\, \min(i,m)-1]$, if present; otherwise $gbc_p(i, t[s+i])$ evaluates to $i+1$.

$$gbc_p(i,c) =_{\mathrm{Def}} \min(\{i-k \mid 0 \leq k < \min(i,m) \text{ and } p[k]=c\} \cup \{i+1\}),$$

An example



A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] = t[s+m-1]$:

Let i_0 be the rightmost position in the substring $p[0\,..\,m-2]$ such that $p[i_0] = p[m-1]$, provided that $p[m-1]$ occur in $p[0\,..\,m-2]$; otherwise let i_0 be -1. Then the occurrence relative position $q_1 = 2m - i_0 - 2$ is safe for shifting.

$$q_1 =_{\text{Def}} \min(\{2m-i-2 \,|\, p[i] = p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}) \,.$$

A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] = t[s+m-1]$:

Let i_0 be the rightmost position in the substring $p[0\,..\,m-2]$ such that $p[i_0] = p[m-1]$, provided that $p[m-1]$ occur in $p[0\,..\,m-2]$; otherwise let i_0 be -1. Then the occurrence relative position $q_1 = 2m - i_0 - 2$ is safe for shifting.

$$q_1 =_{\text{Def}} \min(\{2m-i-2 \mid p[i] = p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\})\,.$$

An example

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

					i_0			
d	c	a	d	b	d	c	a	d

A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] = t[s+m-1]$:

Let i_0 be the rightmost position in the substring $p[0\,..\,m-2]$ such that $p[i_0] = p[m-1]$, provided that $p[m-1]$ occur in $p[0\,..\,m-2]$; otherwise let i_0 be -1. Then the occurrence relative position $q_1 = 2m - i_0 - 2$ is safe for shifting.

$$q_1 =_{\text{Def}} \min(\{2m-i-2 \mid p[i] = p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}).$$

An example

														q_1							
a	b	c	a	b	b	c	b	d	c	a	d	c	b	a	a	d	c	e	a	b	a

					i_0			
d	c	a	d	b	d	c	a	d

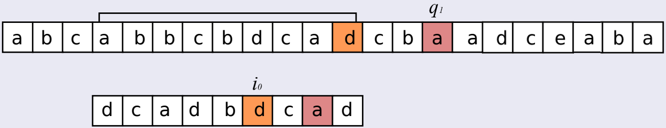
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] = t[s+m-1]$:
Let i_0 be the rightmost position in the substring $p[0\,..\,m-2]$ such that $p[i_0] = p[m-1]$, provided that $p[m-1]$ occur in $p[0\,..\,m-2]$; otherwise let i_0 be -1. Then the occurrence relative position $q_1 = 2m - i_0 - 2$ is safe for shifting.

$$q_1 =_{\text{Def}} \min(\{2m-i-2 \mid p[i] = p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}) .$$

An example



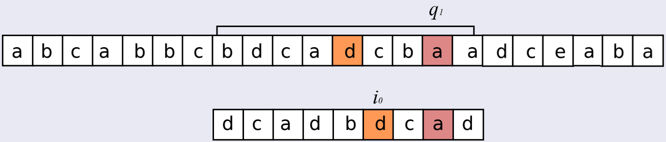
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] = t[s+m-1]$:
Let i_0 be the rightmost position in the substring $p[0\,..\,m-2]$ such that $p[i_0] = p[m-1]$, provided that $p[m-1]$ occur in $p[0\,..\,m-2]$; otherwise let i_0 be -1. Then the occurrence relative position $q_1 = 2m - i_0 - 2$ is safe for shifting.

$$q_1 =_{\text{Def}} \min(\{2m-i-2 \mid p[i] = p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\})\,.$$

An example



A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] \neq t[s+m-1]$:

In this case, let i_0 be the rightmost position in $p[0\,..\,m-2]$ such that $p[i_0] \neq p[m-1]$, provided that $p[0\,..\,m-2]$ contain some character distinct from $p[m-1]$, otherwise let i_0 be -1. Then the occurrence relative position $q_2 = 2m - i_0 - 2$ is safe for shifting.

$$q_2 =_{\text{Def}} \min(\{2m-i-2 \mid p[i] \neq p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}).$$

A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] \neq t[s+m-1]$:

In this case, let i_0 be the rightmost position in $p[0\,..\,m-2]$ such that $p[i_0] \neq p[m-1]$, provided that $p[0\,..\,m-2]$ contain some character distinct from $p[m-1]$, otherwise let i_0 be -1. Then the occurrence relative position $q_2 = 2m - i_0 - 2$ is safe for shifting.

$$q_2 =_{\mathrm{Def}} \min(\{2m-i-2 \mid p[i] \neq p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\})\,.$$

An example

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

				i_0				
d	c	a	d	b	d	d	d	a

A Simple Improved Occurrence Heuristic

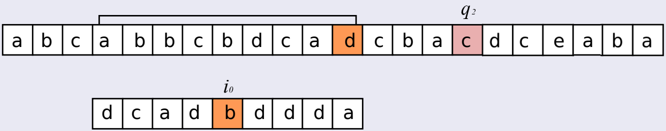
The Improved Occurrence Heuristic

Case $p[m-1] \neq t[s+m-1]$:

In this case, let i_0 be the rightmost position in $p[0\,..\,m-2]$ such that $p[i_0] \neq p[m-1]$, provided that $p[0\,..\,m-2]$ contain some character distinct from $p[m-1]$, otherwise let i_0 be -1. Then the occurrence relative position $q_2 = 2m - i_0 - 2$ is safe for shifting.

$$q_2 =_{\text{Def}} \min(\{2m-i-2 \,|\, p[i] \neq p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}) .$$

An example



A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] \neq t[s+m-1]$:

In this case, let i_0 be the rightmost position in $p[0\,..\,m-2]$ such that $p[i_0] \neq p[m-1]$, provided that $p[0\,..\,m-2]$ contain some character distinct from $p[m-1]$, otherwise let i_0 be -1. Then the occurrence relative position $q_2 = 2m - i_0 - 2$ is safe for shifting.

$$q_2 =_{\text{Def}} \min(\{2m-i-2 \mid p[i] \neq p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}).$$

An example

q_2

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

i_0

d	c	a	d	b	d	d	d	a

A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

Case $p[m-1] \neq t[s+m-1]$:

In this case, let i_0 be the rightmost position in $p[0\,..\,m-2]$ such that $p[i_0] \neq p[m-1]$, provided that $p[0\,..\,m-2]$ contain some character distinct from $p[m-1]$, otherwise let i_0 be -1. Then the occurrence relative position $q_2 = 2m - i_0 - 2$ is safe for shifting.

$$q_2 =_{\text{Def}} \min(\{2m - i - 2 \,|\, p[i] \neq p[m-1] \text{ and } 0 \leq i \leq m-2\} \cup \{2m-1\}) .$$

An example

q_2

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

d	c	a	d	b	d	d	d	a

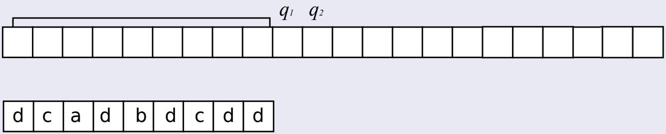
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\text{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\text{Def}} gbc_p(q_2, c)\,.$$

An example



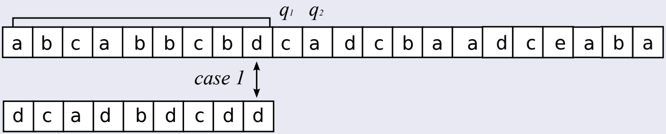
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\mathrm{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\mathrm{Def}} gbc_p(q_2, c)\,.$$

An example



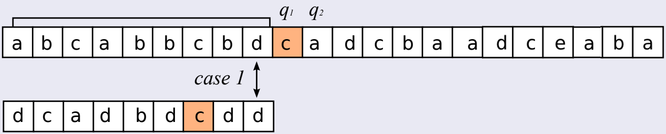
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\text{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\text{Def}} gbc_p(q_2, c)\,.$$

An example



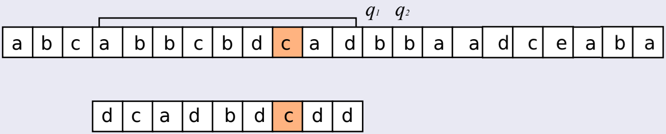
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\text{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\text{Def}} gbc_p(q_2, c)\,.$$

An example



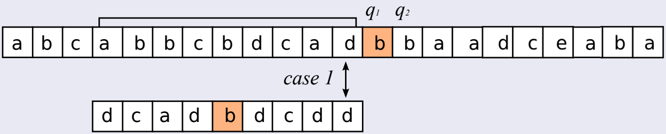
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\text{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\text{Def}} gbc_p(q_2, c)\,.$$

An example



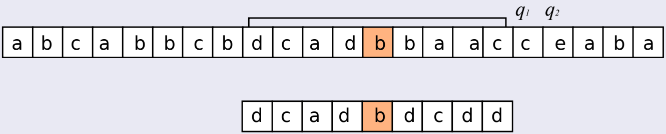
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\mathrm{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\mathrm{Def}} gbc_p(q_2, c)\,.$$

An example



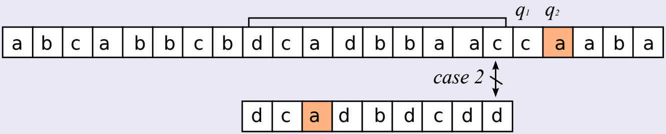
A Simple Improved Occurrence Heuristic

The Improved Occurrence Heuristic

The two occurrence relative positions q_1 and q_2 are then used by our Improved Occurrence Heuristic to calculate the shift advancements during the searching phase of the algorithm:

$$ibc1_p(c) =_{\text{Def}} gbc_p(q_1, c)\,, \qquad ibc2_p(c) =_{\text{Def}} gbc_p(q_2, c)\,.$$

An example



A Simple Improved Occurrence Heuristic

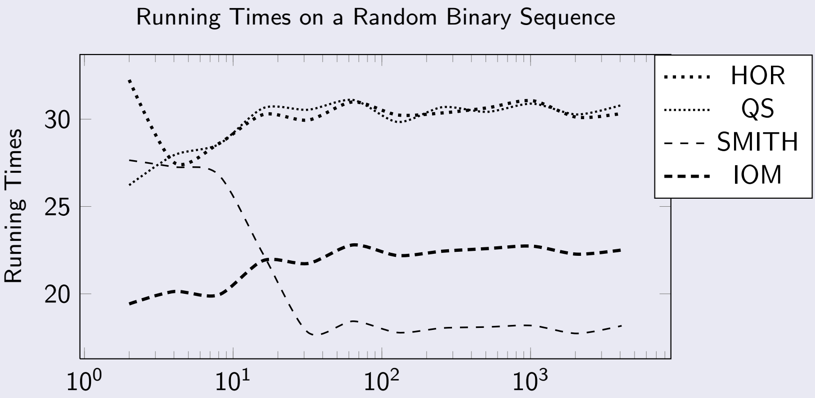
The Improved Occurrence Matcher

```
PrecomputeIOH(p, m, step)
    for each c ∈ Σ do
        ibc[c] ← step + 1
    for i ← 0 to m − 1 do
        ibc[p[i]] ← step − i
    return ibc
```

```
ImprovedOccurrenceMatcher(p, m, t, n)
    step1 ← step2 ← 2m − 1
    for i ← 0 to m − 2 do
        if p[i] = p[m − 1] then
        then step1 ← 2m − i − 2
        else step2 ← 2m − i − 2
    ibc1 ← PrecomputeIOH(p, m, step1)
    ibc2 ← PrecomputeIOH(p, m, step2)
    s ← 0
    while (s ≤ n − m) do
        if (p[m − 1] = t[s + m − 1]) then
            i ← 0
            while (i < m and p[i] = t[s + i]) do
                i ← i + 1
            if (i = m) then Output(s)
            s ← s + ibc1[t[s + step1]]
        else s ← s + ibc2[t[s + step2]]
```

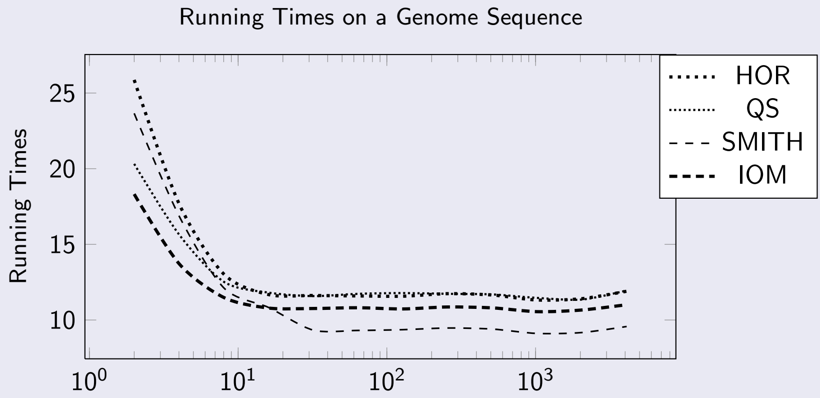
A Simple Improved Occurrence Heuristic

The Improved Occurrence Matcher



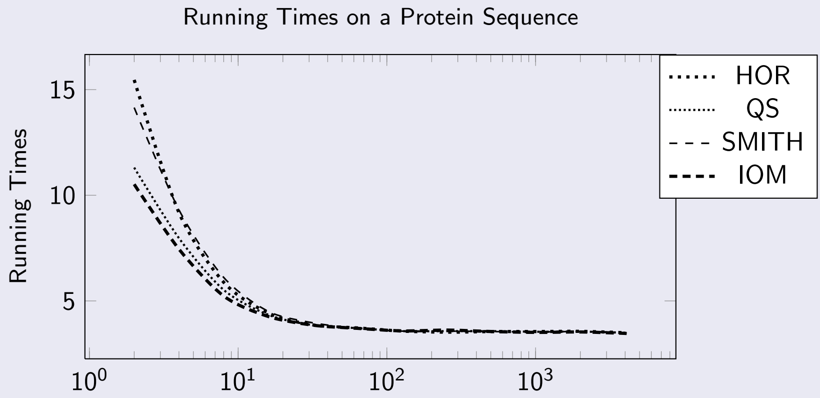
A Simple Improved Occurrence Heuristic

The Improved Occurrence Matcher



A Simple Improved Occurrence Heuristic

The Improved Occurrence Matcher



A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c) .$$

A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\mathrm{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c) .$$

An example

P

d	c	a	d	b	a	c	a	d

frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c) \, .$$

An example

P

					i			
d	c	a	d	b	a	c	a	d

frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

$$adv(i) = f(a)\ gbc(i,a) + f(b)\ gbc(i,b) + f(c)\ gbc(i,c) + f(d)\ gbc(i,d)$$

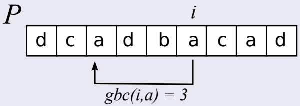
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c) \, .$$

An example



frequency

$f(a) = 0.5$

$f(b) = 0.25$

$f(c) = 0.15$

$f(d) = 0.1$

$adv(i) = 0.5 \times 3 + f(b)\, gbc(i,b) + f(c)\, gbc(i,c) + f(d)\, gbc(i,d)$

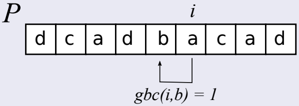
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i,c)\,.$$

An example



frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

$adv(i) = 0.5 \times 3 + 0.25 \times 1 + f(c)\ gbc(i,c) + f(d)\ gbc(i,d)$

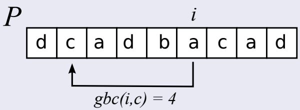
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c)\,.$$

An example



frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

$adv(i) = 0.5 \times 3 + 0.25 \times 1 + 0.15 \times 4 + f(d)\, gbc(i,d)$

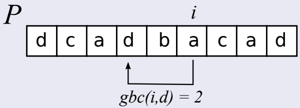
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c)\,.$$

An example



frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

$adv(i) = 0.5 \times 3 + 0.25 \times 1 + 0.15 \times 4 + 0.1 \times 2$

A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

For a given occurrence relative position $0 \leq i \leq m$, the average shift advancement of the generalized occurrence function gbc_p is given by the function

$$adv_{p,f}(i) =_{\text{Def}} \sum_{c \in \Sigma} f(c) \cdot gbc_p(i, c)\,.$$

An example

P

					i			
d	c	a	d	b	a	c	a	d

frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

$adv(i) = 2.55$

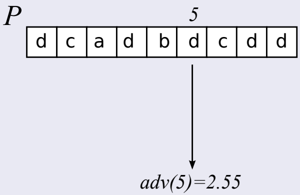
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

We then define the *worst-occurrence relative position* q^* as the smallest position $0 \leq q \leq m$ which maximizes $adv_{p,f}(q)$, i.e.,

$$q^* =_{\text{Def}} \min\{q \mid 0 \leq q \leq m \text{ and } adv_{p,f}(q) = \max_{0 \leq i \leq m} adv_{p,f}(i)\}\,.$$

An example



frequency

f(a) = 0.5
f(b) = 0.25
f(c) = 0.15
f(d) = 0.1

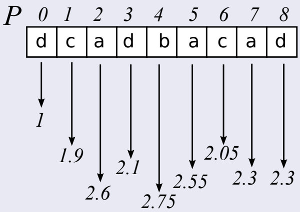
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

We then define the *worst-occurrence relative position* q^* as the smallest position $0 \leq q \leq m$ which maximizes $adv_{p,f}(q)$, i.e.,

$$q^* =_{\text{Def}} \min\{q \mid 0 \leq q \leq m \text{ and } adv_{p,f}(q) = \max_{0 \leq i \leq m} adv_{p,f}(i)\}\,.$$

An example



frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

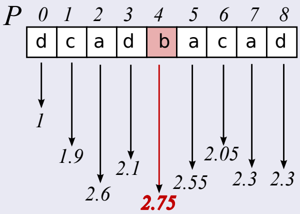
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

We then define the *worst-occurrence relative position* q^* as the smallest position $0 \leq q \leq m$ which maximizes $adv_{p,f}(q)$, i.e.,

$$q^* =_{\text{Def}} \min\{q \mid 0 \leq q \leq m \text{ and } adv_{p,f}(q) = \max_{0 \leq i \leq m} adv_{p,f}(i)\} .$$

An example



frequency

$f(a) = 0.5$
$f(b) = 0.25$
$f(c) = 0.15$
$f(d) = 0.1$

A Self Tuned Occurrence Heuristic

Finding the relative frequency of characters

(1) In a preprocessing phase, compute the character frequencies of an initial segment of the text (say of no more than γ characters).

A Self Tuned Occurrence Heuristic

Finding the relative frequency of characters

(1) In a preprocessing phase, compute the character frequencies of an initial segment of the text (say of no more than γ characters).

(2) Run the first γ iterations of the algorithm, assuming *a priori* a default distribution of characters (e.g., the uniform distribution). At the same time, compute the relative frequency of the first γ characters and then recompute the occurrence heuristic according to the estimated frequency.

A Self Tuned Occurrence Heuristic

Finding the relative frequency of characters

(1) In a preprocessing phase, compute the character frequencies of an initial segment of the text (say of no more than γ characters).

(2) Run the first γ iterations of the algorithm, assuming *a priori* a default distribution of characters (e.g., the uniform distribution). At the same time, compute the relative frequency of the first γ characters and then recompute the occurrence heuristic according to the estimated frequency.

(3) While running the algorithm, keep updating the relative frequencies of the characters. At regular intervals (say of γ characters), or when the difference between the current relative frequencies and the one used in the worst-occurrence heuristic exceeds a threshold, recompute the heuristic.

A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic

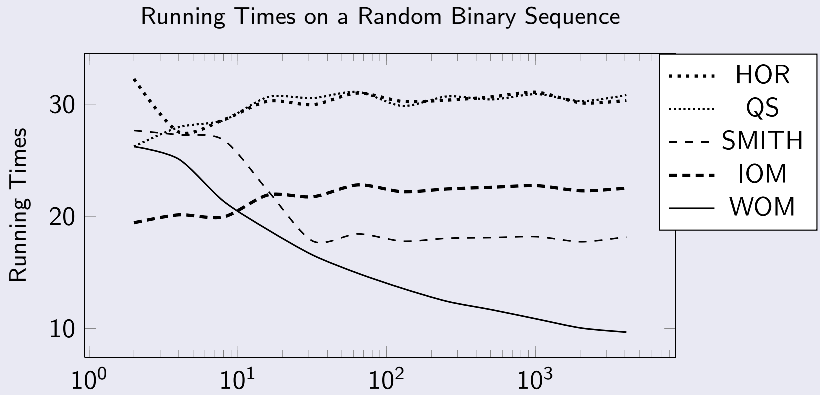
```
FindWorstOccurrence(p, m, Σ, f)
    for each c ∈ Σ do
        lp[c] ← −1
    adv ← 1
    max ← 1
    q ← 0
    for i ← 1 to m do
        gbc ← i − lp[p[i − 1]] − 1
        adv ← adv − f(p[i − 1]) · gbc + 1
        lp[p[i − 1]] ← i − 1
        if (adv > max) then
            max ← adv
            q ← i
    return q
```

```
PrecomputeWOH(p, m, q)
    for each c ∈ Σ do
        wo[c] ← q + 1
    for i ← 0 to q − 1 do
        wo[p[i]] ← q − i
    return wo

WorstOccurrenceMatcher(p, m, t, n)
    q ← FindWorstOccurrence(p, m, Σ, f)
    wo ← PrecomputeWOH(p, m, q)
    s ← 0
    while (s ≤ n − m) do
        i ← 0
        while (i < m and p[i] = t[s + i]) do
            i ← i + 1
        if (i = m) then Output(s)
        s ← s + wo[t[s + q]]
```

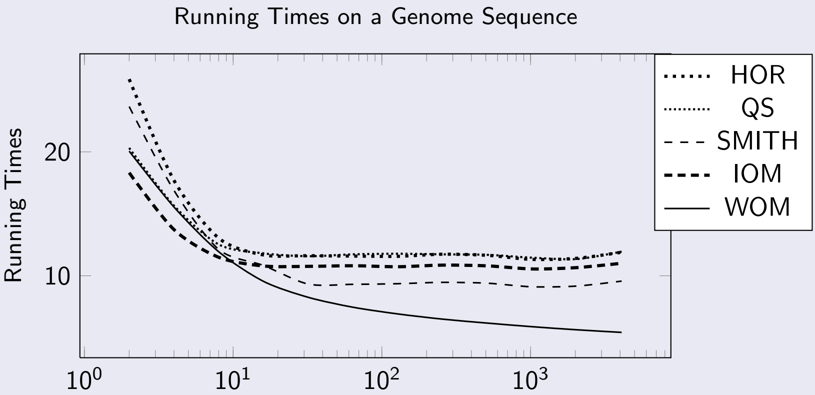
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic



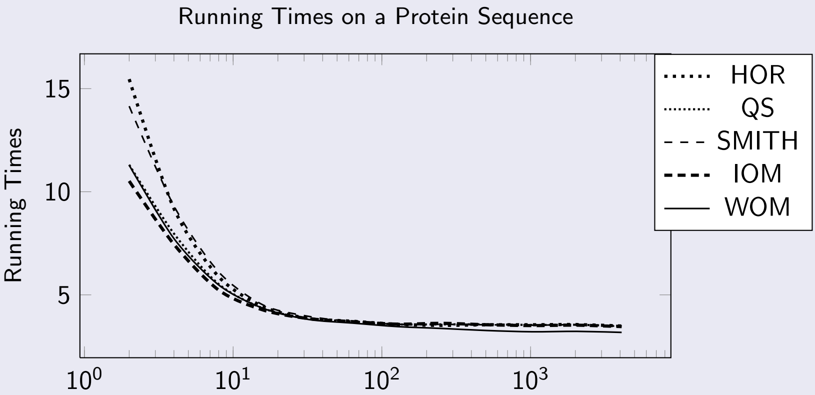
A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic



A Self Tuned Occurrence Heuristic

A Self Tuned Occurrence Heuristic



The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

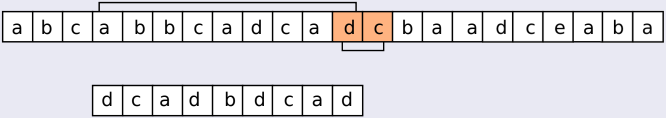
Some algorithms use an occurrence heuristic based on two consecutive characters. In such cases the distance between the two characters involved in the occurrence heuristics is 1. However it may be possible that other occurrence jump distances generate larger shift advancements.

The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

Some algorithms use an occurrence heuristic based on two consecutive characters. In such cases the distance between the two characters involved in the occurrence heuristics is 1. However it may be possible that other occurrence jump distances generate larger shift advancements.

An example

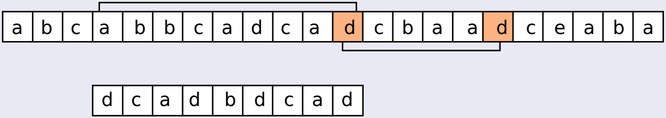


The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

Some algorithms use an occurrence heuristic based on two consecutive characters. In such cases the distance between the two characters involved in the occurrence heuristics is 1. However it may be possible that other occurrence jump distances generate larger shift advancements.

An example

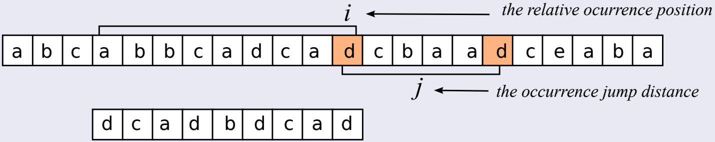


The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

i is the *relative occurrence position*, the position in the text of the first character involved in the computation of the occurrence heuristic;
j is the *occurrence jump distance*, the distance between the two characters involved in the computation of the occurrence heuristic.

An example



The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

We introduce the *generalized double occurrence function* $gbc_p^2(i, j, c_1, c_2)$ *relative to* p, with $0 \leq i \leq m$, $1 \leq j \leq m$ and $c_1, c_2 \in \Sigma$, intended to calculate the largest *safe* shift advancement for p compatible with the constraints $t[s+i] = c_1$ and $t[s+i+j] = c_2$, when p has shift s with respect to a text t.

The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

We introduce the *generalized double occurrence function* $gbc_p^2(i, j, c_1, c_2)$ *relative to* p, with $0 \leq i \leq m$, $1 \leq j \leq m$ and $c_1, c_2 \in \Sigma$, intended to calculate the largest *safe* shift advancement for p compatible with the constraints $t[s+i] = c_1$ and $t[s+i+j] = c_2$, when p has shift s with respect to a text t.

Mathematical Definition

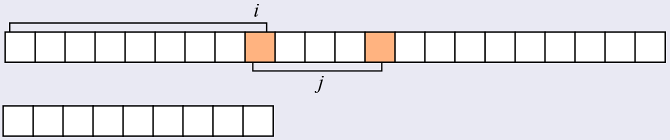
$$\begin{aligned} gbc_p^2(i, j, c_1, c_2) =_{\mathrm{Def}} \min(\quad & \{i - k \mid m - j \leq k < i \wedge p[k] = c_1\} \\ & \cup \{i - k \mid 0 \leq k < \min(m - j, i) \wedge p[k] = c_1 \wedge p[k+j] = c_2\} \\ & \cup \{i + j - k \mid 0 \leq k < j \wedge p[k] = c_2\} \\ & \cup \{i + j + 1\} \ ). \end{aligned}$$

The Jumping Occurrence Heuristic

Mathematical Definition

$$
\begin{aligned}
gbc_p^2(i,j,c_1,c_2) =_{\mathrm{Def}} \min(\quad & \{i-k \mid m-j \le k < i \wedge p[k]=c_1\} \\
& \cup \{i-k \mid 0 \le k < \min(m-j,i) \wedge p[k]=c_1 \wedge p[k+j]=c_2\} \\
& \cup \{i+j-k \mid 0 \le k < j \wedge p[k]=c_2\} \\
& \cup \{i+j+1\}\ ).
\end{aligned}
$$

An example

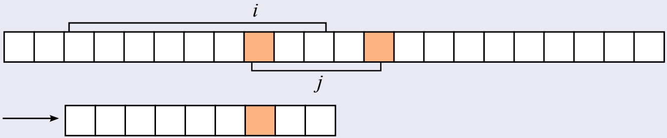


The Jumping Occurrence Heuristic

Mathematical Definition

$$
\begin{aligned}
gbc_p^2(i, j, c_1, c_2) =_{\text{Def}} \min(\quad & \{i - k \mid m - j \leq k < i \wedge p[k] = c_1\} \\
& \cup \{i - k \mid 0 \leq k < \min(m - j, i) \wedge p[k] = c_1 \wedge p[k + j] = c_2\} \\
& \cup \{i + j - k \mid 0 \leq k < j \wedge p[k] = c_2\} \\
& \cup \{i + j + 1\} \;).
\end{aligned}
$$

An example

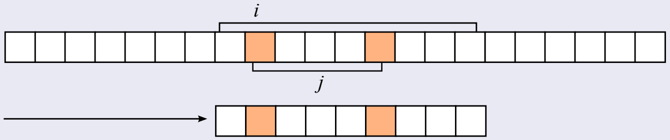


The Jumping Occurrence Heuristic

Mathematical Definition

$$
\begin{aligned}
gbc_p^2(i, j, c_1, c_2) =_{\text{Def}} \min(\quad & \{i - k \mid m - j \leq k < i \wedge p[k] = c_1\} \\
& \cup \{i - k \mid 0 \leq k < \min(m - j, i) \wedge p[k] = c_1 \wedge p[k + j] = c_2\} \\
& \cup \{i + j - k \mid 0 \leq k < j \wedge p[k] = c_2\} \\
& \cup \{i + j + 1\} \ ).
\end{aligned}
$$

An example

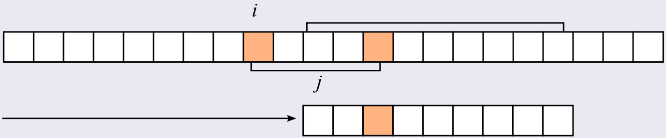


The Jumping Occurrence Heuristic

Mathematical Definition

$$
\begin{aligned}
gbc_p^2(i, j, c_1, c_2) =_{\text{Def}} \min(\quad & \{i - k \mid m - j \leq k < i \wedge p[k] = c_1\} \\
& \cup \{i - k \mid 0 \leq k < \min(m - j, i) \wedge p[k] = c_1 \wedge p[k + j] = c_2\} \\
& \cup \{i + j - k \mid 0 \leq k < j \wedge p[k] = c_2\} \\
& \cup \{i + j + 1\} \ ).
\end{aligned}
$$

An example

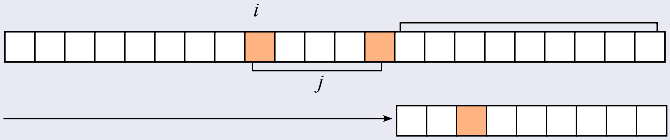


The Jumping Occurrence Heuristic

Mathematical Definition

$$
\begin{aligned}
gbc_p^2(i, j, c_1, c_2) =_{\text{Def}} \min(\quad & \{i - k \mid m - j \leq k < i \wedge p[k] = c_1\} \\
& \cup \{i - k \mid 0 \leq k < \min(m - j, i) \wedge p[k] = c_1 \wedge p[k + j] = c_2\} \\
& \cup \{i + j - k \mid 0 \leq k < j \wedge p[k] = c_2\} \\
& \cup \{i + j + 1\} \quad).
\end{aligned}
$$

An example



The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

Let j be a fixed relative jump distance to be used by the generalized double occurrence function gbc_p^2 with relative occurrence position i. In order for the character $t[s+i+j]$ to be involved in the computation of the advancement by gbc_p^2, we must have

$$gbc_p(i, t[s+i]) \geq j$$

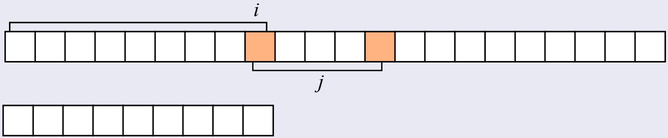
The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

Let j be a fixed relative jump distance to be used by the generalized double occurrence function gbc_p^2 with relative occurrence position i. In order for the character $t[s+i+j]$ to be involved in the computation of the advancement by gbc_p^2, we must have

$$gbc_p(i, t[s+i]) \geq j$$

An example



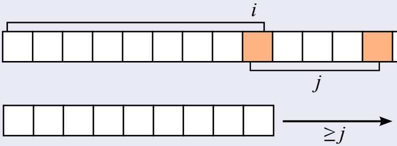
The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

Let j be a fixed relative jump distance to be used by the generalized double occurrence function gbc_p^2 with relative occurrence position i. In order for the character $t[s+i+j]$ to be involved in the computation of the advancement by gbc_p^2, we must have

$$gbc_p(m-1, t[s+i]) \geq j$$

An example



The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

Thus, for a fixed bound $0 \leq \beta \leq 1$, the computation of the shift advancement will involve the second character with a probability of at least β if and only if its jump distance j satisfies

$$Pr\{gbc_p(i, c) \geq j \mid c \in \Sigma\} \geq \beta\,.$$

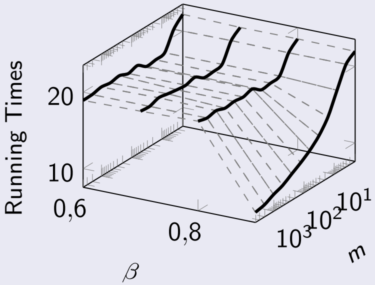
This suggests to use the following relative jump distance

$$j^*_\beta =_{\mathrm{Def}} \max\left\{\ell \mid 1 \leq \ell \leq m \text{ and } Pr\{gbc_p(i, c) \geq \ell \mid c \in \Sigma\} \geq \beta\right\}$$

The Jumping Occurrence Heuristic

The Jumping Occurrence Heuristic

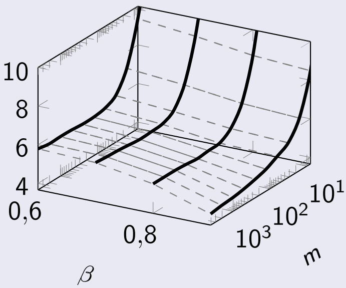
Running Times on a Random Binary Sequence



The Jumping Occurrence Heuristic

The Jumping Occurrence Heuristic

Running Times on a Genome Sequence



The Jumping Occurrence Heuristic

The Jumping Occurrence Matcher

```
PRECOMPUTEJOH(p, m, i, j)
    for each a ∈ Σ do
        for each b ∈ Σ do
            jbc(a, b) ← i + 1 + j
    for each a ∈ Σ do
        for k ← 0 to j − 1 do
            jbc(a, p[k]) ← i + 1 + j − 1 − k
    for k ← 0 to i + 1 − j − 1 do
            jbc(p[k], p[k + len]) ← i + 1 − 1 − k
    for k ← i + 1 − j to m − 1 do
        for each a ∈ Σ do
            jbc(p[k], a) ← i + 1 − 1 − j
```

The Jumping Occurrence Heuristic

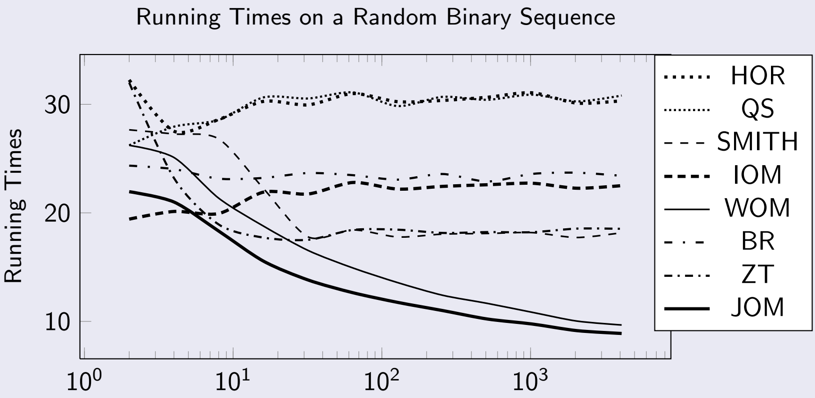
The Jumping Occurrence Matcher

```
FindJumpDistance(p, m, i, Σ, f, β)
    for each c ∈ Σ do v[c] ← 1
    frq ← j ← 1
    while (frq ≥ β and j ≤ i + 1) do
        if (v[p[i + 1 − j]] = 1) then
            v[p[i + 1 − j]] = 0
            frq ← frq − f(p[i + 1 − j])
        j ← j + 1
    return j − 1

JumpingOccurrenceMatcher(p, m, t, n)
    i ← FindWorstOccurrence(p, m, Σ, f)
    j ← FindJumpDistance(p, m, i, Σ, f, 0.9)
    jbc ← PrecomputeJOH(p, m, i, j)
    s ← 0
    while (s ≤ n − m) do
        k ← 0
        while (k < m and p[k] = t[s + k]) do  k ← k + 1
        if (k = m) then Output(s)
        s ← s + jbc(t[s + i], t[s + i + j])
```

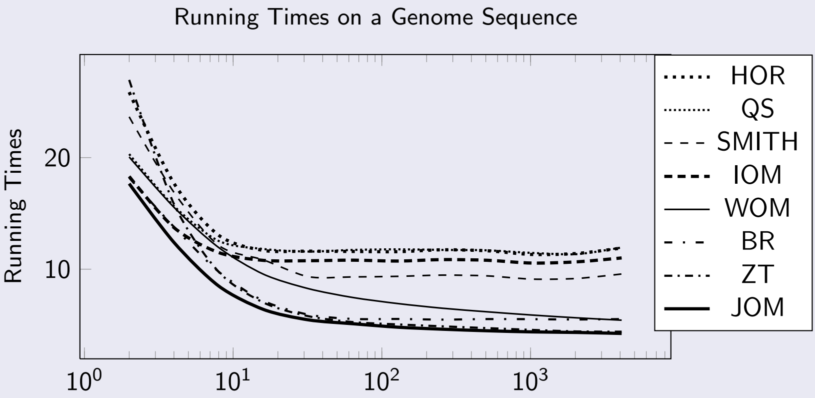
The Jumping Occurrence Heuristic

The Jumping Occurrence Heuristic



The Jumping Occurrence Heuristic

The Jumping Occurrence Heuristic



The Jumping Occurrence Heuristic

The Jumping Occurrence Heuristic