

Data Structures and Algorithms

— Lecture 12: Text Processing —

Jörg Endrullis

13th of October

Overview

Pattern Matching

Boyer-Moore Algorithm

Knuth-Morris-Pratt Algorithm

Huffman code

Tries

Strings

- ▶ **string**: a sequence of characters
example: abracadabra

Strings

- ▶ **string**: a sequence of characters
example: abracadabra

- ▶ **substring** of S : string of the form $S[i \dots j]$
example: cada is a substring of abra**cad**abra

Strings

- ▶ **string**: a sequence of characters
example: abracadabra
- ▶ **substring** of S : string of the form $S[i \dots j]$
example: cada is a substring of abra**cad**abra
- ▶ **prefix**: substring $S[0 \dots i]$ that begins at the front
example: abr is a prefix of **abr**acadabra

Strings

- ▶ **string**: a sequence of characters
example: abracadabra
- ▶ **substring** of S : string of the form $S[i \dots j]$
example: cada is a substring of abra**cad**abra
- ▶ **prefix**: substring $S[0 \dots i]$ that begins at the front
example: abr is a prefix of **abr**acadabra
- ▶ **suffix**: substring $S[i \dots (\text{length } S - 1)]$ that ends at the end
example: ra is a suffix of abra**cadabra**

Pattern Matching

The **pattern matching problem**:

- ▶ given: text T and pattern P
- ▶ task: find substring P in T

Pattern Matching

The **pattern matching problem**:

- ▶ given: text T and pattern P
- ▶ task: find substring P in T

Example:

- ▶ find $P = \text{dab}$ in $T = \text{adacadabra}$

Pattern Matching

The **pattern matching problem**:

- ▶ given: text T and pattern P
- ▶ task: find substring P in T

Example:

- ▶ find $P = \text{dab}$ in $T = \text{adacada**bra**}$

Pattern Matching

The **pattern matching problem**:

- ▶ given: text T and pattern P
- ▶ task: find substring P in T

Example:

- ▶ find $P = \text{dab}$ in $T = \text{adacadabra}$

Applications:

- ▶ text editors
- ▶ search engines
- ▶ biological reasearch
- ▶ ...

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
1	2	3							
a	b	a							

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

4

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

4

a	b	a
---	---	---

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

4

a	b	a
---	---	---

5

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

4

a	b	a
---	---	---

5

a	b	a
---	---	---

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

4

a	b	a
---	---	---

5

a	b	a
---	---	---

6

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a	b	c	a	c	a	b	a	b	c
---	---	---	---	---	---	---	---	---	---

1 2 3

a	b	a
---	---	---

4

a	b	a
---	---	---

5

a	b	a
---	---	---

6 7

a	b	a
---	---	---

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a b c a c a b a b c

1 2 3

a b a a b a

4

a b a

5

a b a

6 7

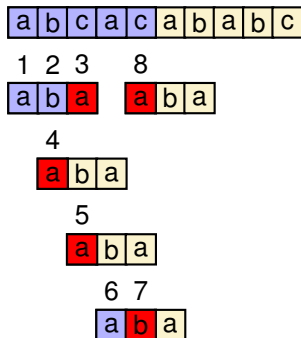
a b a

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in



Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

a b c a c a b a b c

1 2 3 8

a b a a b a

4

a b a a b a

5

a b a

6 7

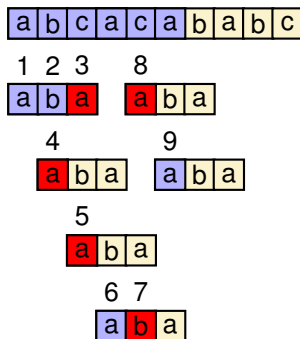
a b a

Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

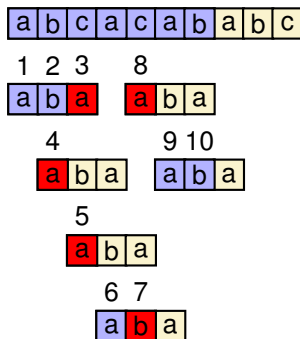


Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in

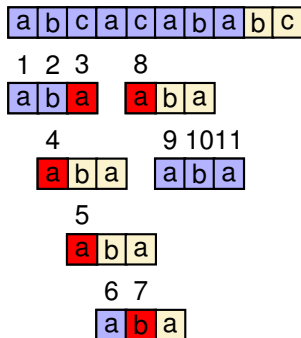


Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

- ▶ for every position $0 \leq p \leq n - m$:
compare P with $T[p, \dots, p + m - 1]$

Example: find $P = aba$ in



Brute-Force Algorithm

Walk from left to right with P of length m through T of length n .

`bruteForceMatch(T, P):`

for $pos = 0$ **to** $n - m$ **do**

$match = \mathbf{true}$

for $i = 0$ **to** $m - 1$ **do**

if $P[i] \neq T[pos + i]$ **then**

$match = \mathbf{false}$

 break the for- i -loop

if $match$ **then return** pos (match at position pos)

return -1 (no match)

Brute-Force Algorithm

Example for worst-case:

- ▶ $T = aaaaaa \dots aaaaab$ and $P = aaaab$

Brute-Force Algorithm

Example for worst-case:

- ▶ $T = aaaaaa \dots aaaaab$ and $P = aaaab$

Then: outer loop $n - m + 1$ times, inner loop m times.

Brute-Force Algorithm

Example for worst-case:

- ▶ $T = aaaaaa \dots aaaaab$ and $P = aaaab$

Then: outer loop $n - m + 1$ times, inner loop m times.

Worst-case time complexity:

- ▶ $O(n \cdot m)$
(length of text times length of pattern)

Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

Example: we search the pattern $P = \text{rithm}$ in

a	p	a	t	t	e	r	n	m	a	t	c	h	i	n	g	a	l	g	o	r	i	t	h	m
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

Example: we search the pattern $P = \text{rithm}$ in

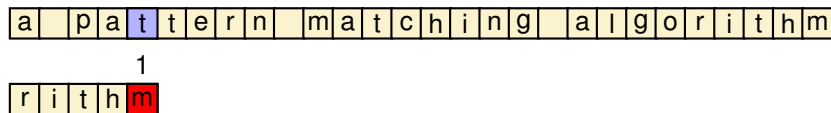
a	p	a	t	t	e	r	n	m	a	t	c	h	i	n	g	a	l	g	o	r	i	t	h	m
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

r	i	t	h	m
---	---	---	---	---

Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

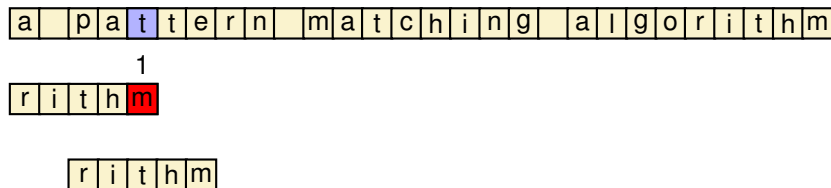
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

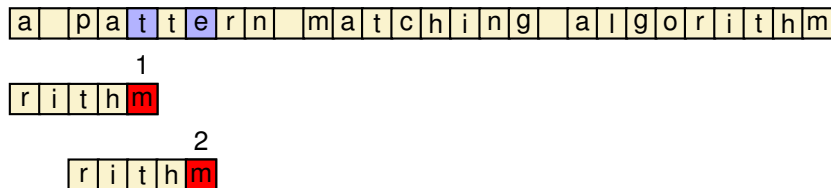
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

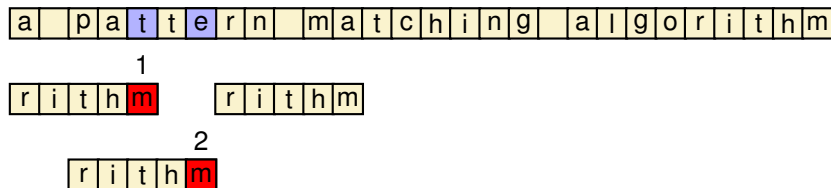
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

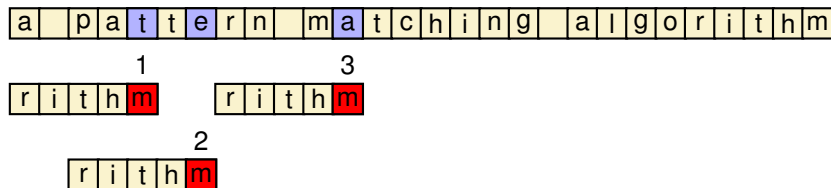
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

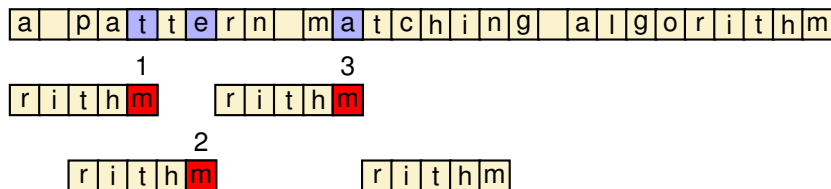
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

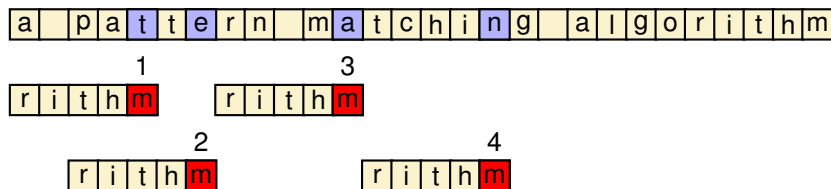
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

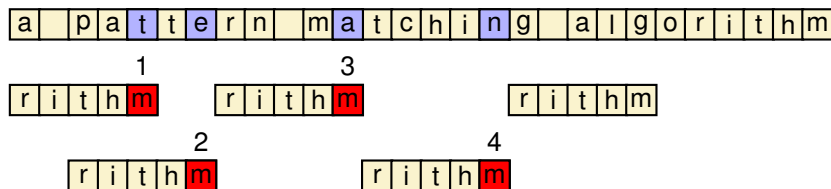
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

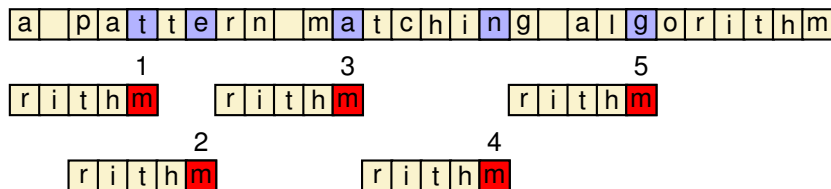
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

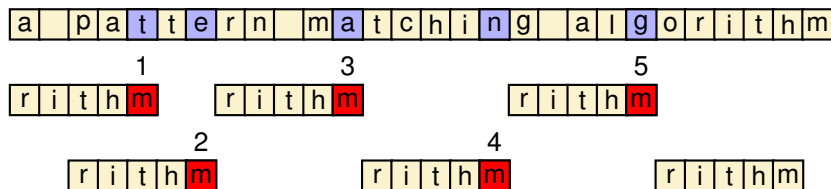
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

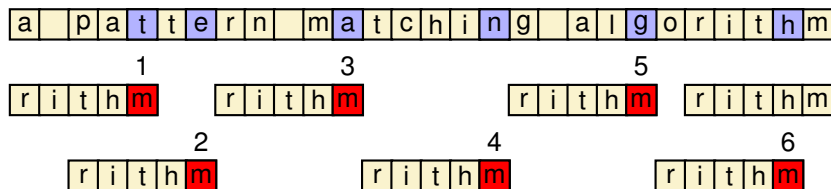
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

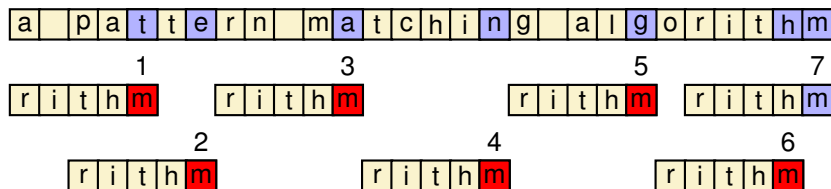
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

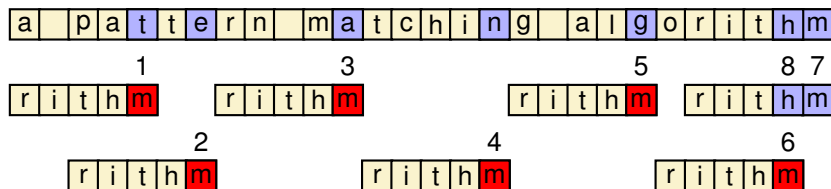
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

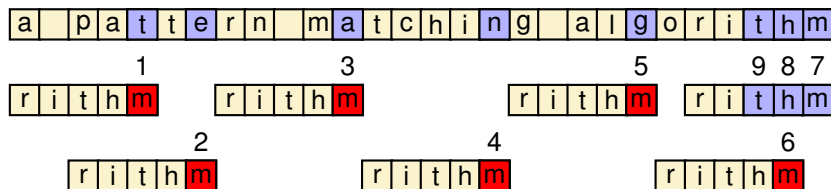
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

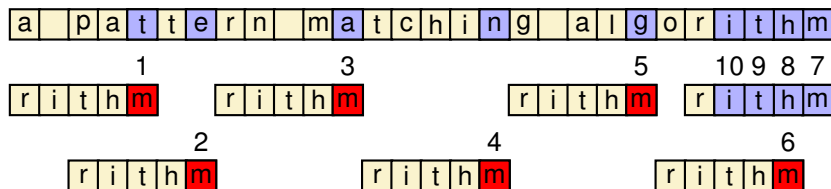
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

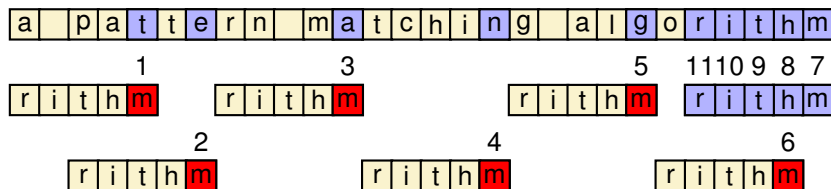
Example: we search the pattern $P = \text{rithm}$ in



Boyer-Moore Algorithm: Idea

- ▶ **compare P with substring of T backwards**
- ▶ **shift if mismatch** at $T[pos] = c$
 - if c not in P : shift begin of P to $T[pos + 1]$
 - if c in P : shift last c in P to $T[pos]$
(instead of negative shift, shift 1 forward)

Example: we search the pattern $P = \text{rithm}$ in



Last-Occurrence Function

The last-occurrence function $L(c)$:

- ▶ **often given as array**

example with $\Sigma = \{a, b, c, d\}$ and $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

- ▶ $L(c)$ is **index of last occurrence** of c in P
and -1 if c does not occur in P

Last-Occurrence Function

The last-occurrence function $L(c)$:

- ▶ **often given as array**

example with $\Sigma = \{a, b, c, d\}$ and $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

- ▶ $L(c)$ is **index of last occurrence** of c in P and -1 if c does not occur in P
- ▶ **time complexity** computing last-occurrence array:
 $O(m + s)$
where m is the length of P and s the size of the alphabet

Boyer-Moore Algorithm

BoyerMooreMatch(T, P, Σ):

$L = \text{lastOccurrenceFunction}(P, \Sigma)$

$pos = 0$

while $pos \leq n - m$ **do**

$match = \text{true}$

for $i = m - 1$ **downto** 0 **do**

if $P[i] \neq T[pos + i]$ **then**

$match = \text{false}$

(align last occurrence, but move at least 1)

$pos = pos + \max(1, i - L(T[pos + i]))$

break the for- i -loop

done

if $match$ **then return** pos (match at position pos)

done

return -1 (no match)

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

a	b	a	c	a	a	b	c	a	d	a	b	a	c	a	b	a	a	b	b
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

a	b	a	c	a	b
0	1	2	3	4	5

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b
0 1 2 3 4 5

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b
0 1 2 3 4 5

a b a c a b
0 1 2 3 4 5

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b
0 1 2 3 4 5

2
a b a c a b
0 1 2 3 4 5

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b

0 1 2 3 4 5
3 2
a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b

0 1 2 3 4 5
4 3 2
a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b

4 3 2
a b a c a b

a b a c a b

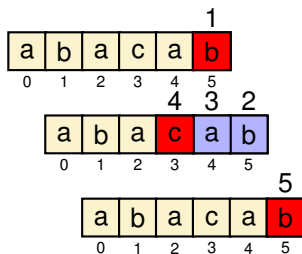
Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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



Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b

4 3 2
a b a c a b

5
a b a c a b

a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

c	a	b	c	d
$L(c)$	4	5	3	-1

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

1
a b a c a b

4 3 2
a b a c a b

5
a b a c a b

6
a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	

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

1
a b a c a b

4 3 2
a b a c a b

5
a b a c a b

a b a c a b

6
a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	

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

1
a b a c a b
0 1 2 3 4 5

4 3 2
a b a c a b
0 1 2 3 4 5

5
a b a c a b
0 1 2 3 4 5

7
a b a c a b
0 1 2 3 4 5

6
a b a c a b
0 1 2 3 4 5

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	

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

1
a b a c a b

4 3 2
a b a c a b

5
a b a c a b

8 7
a b a c a b

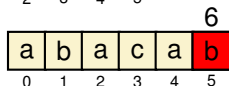
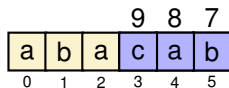
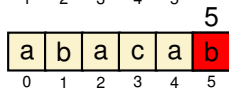
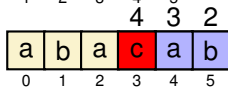
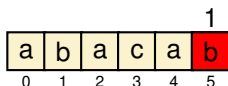
6
a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	



Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	

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

1
a b a c a b

4 3 2
a b a c a b

5
a b a c a b

10 9 8 7
a b a c a b

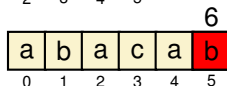
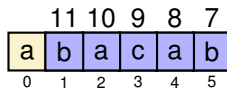
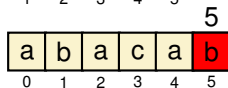
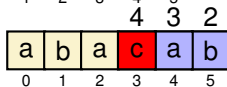
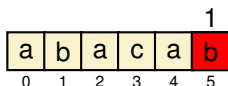
6
a b a c a b

Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	



Example

Example:

- ▶ $\Sigma = \{a, b, c, d\}$
- ▶ $P = abacab$

	c	a	b	c	d
$L(c)$	4	5	3	-1	

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

1
a b a c a b
0 1 2 3 4 5

4 3 2
a b a c a b
0 1 2 3 4 5

5
a b a c a b
0 1 2 3 4 5

12 11 10 9 8 7
a b a c a b
0 1 2 3 4 5

6
a b a c a b
0 1 2 3 4 5

Boyer-Moore: Analysis

Example for worst-case:

Boyer-Moore: Analysis

Example for worst-case:

- ▶ $T = \text{aaaaa} \dots \text{aaaaa}$ and $P = \text{baaaaa}$

Boyer-Moore: Analysis

Example for worst-case:

- ▶ $T = \text{aaaaa} \dots \text{aaaaa}$ and $P = \text{baaaaa}$

The worst-case is:

- ▶ unlikely in English text
- ▶ may occur in images or DNA sequences

Boyer-Moore: Analysis

Example for worst-case:

- ▶ $T = \text{aaaaa} \dots \text{aaaaa}$ and $P = \text{baaaaa}$

The worst-case is:

- ▶ unlikely in English text
- ▶ may occur in images or DNA sequences

The **worst-case time complexity**:

- ▶ $O(m + s) + O(n \cdot m) = O(n \cdot m + s)$
(n length of T , m length of P and s size of the alphabet)

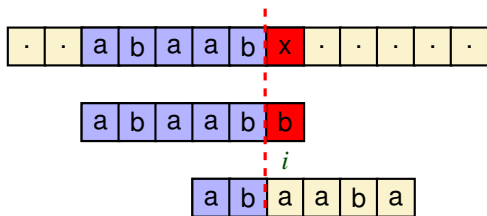
Boyer-Moore's algorithm is

- ▶ significantly faster than brute-force for English text

Knuth-Morris-Pratt Algorithm: Idea

- ▶ compares the pattern left-to-right
- ▶ if mismatch $T[pos] \neq P[i]$: what is the largest shift?

largest prefix of $P[0 \dots i-1]$ that is suffix of $P[1 \dots i-1]$



No need to compare again.

Resume comparing here.

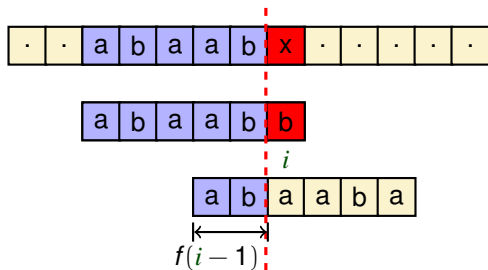
KMP Failure (Shift) Function

The Knuth-Morris-Pratt's **failure function** $f(i)$:

- ▶ compute $f(i)$ for every position in P
- ▶ $f(i)$ is the largest j s.t. $P[0 \dots j - 1]$ is suffix of $P[1 \dots i]$

Example: $P = \text{abaaba}$

i	0	1	2	3	4	5
$P[i]$	a	b	a	a	b	a
$f(i)$	0	0	1	1	2	3



Knuth-Morris-Pratt Algorithm

KMPMatch(T, P):

$F = \text{failureFunction}(P)$

$pos = 0$

$i = 0$

while $pos \leq n - m$ **do**

if $P[i] \neq T[pos + i]$ **then**

if $i > 0$ **then**

$pos = pos + i - f(i - 1)$

$i = f(i - 1)$

else

$pos = pos + 1$

else

$i = i + 1$

if $i == m$ **then return** pos (match at position pos)

done

return -1 (no match)

Example

Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.

a	b	a	a	b	c	a	b	a	b	a	a	b	a	a	b	a	a	b	b
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.

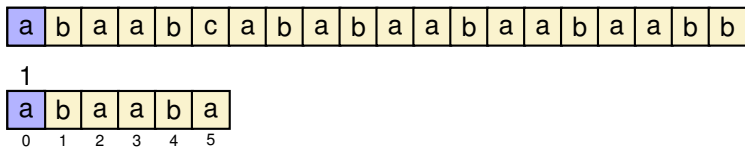
a	b	a	a	b	c	a	b	a	b	a	a	b	a	a	b	a	a	b	b
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

a	b	a	a	b	a
0	1	2	3	4	5

i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

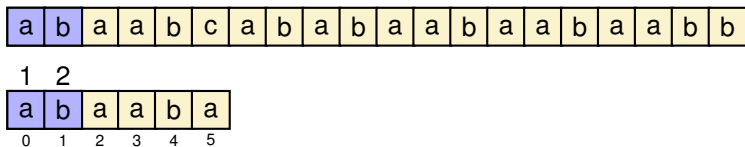
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

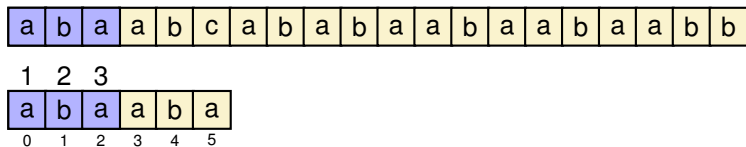
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

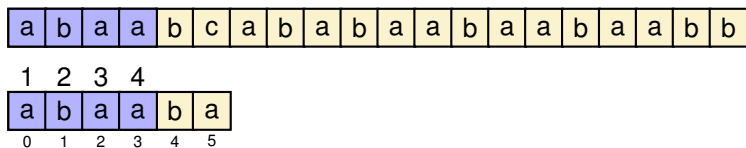
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

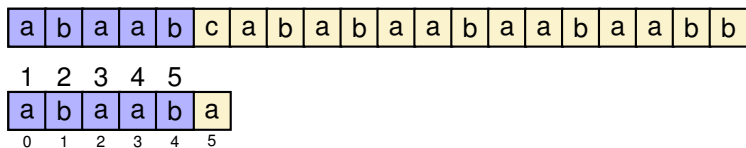
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

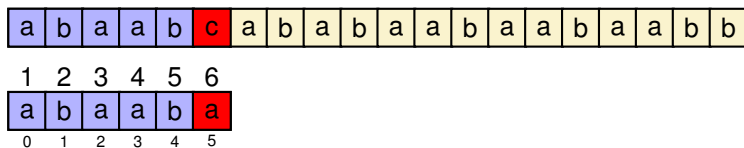
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

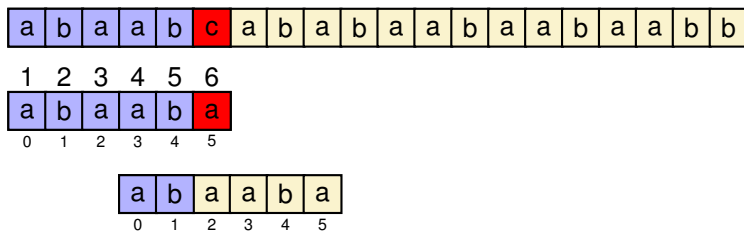
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

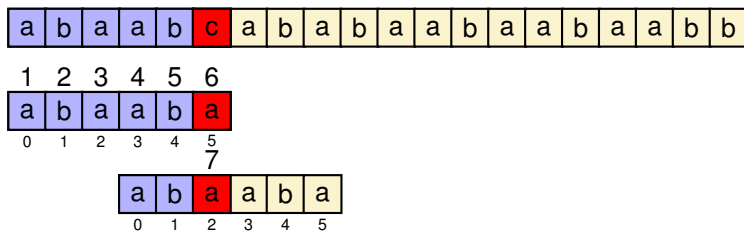
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

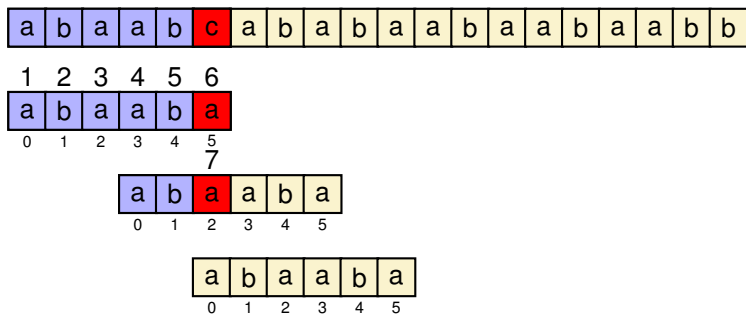
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

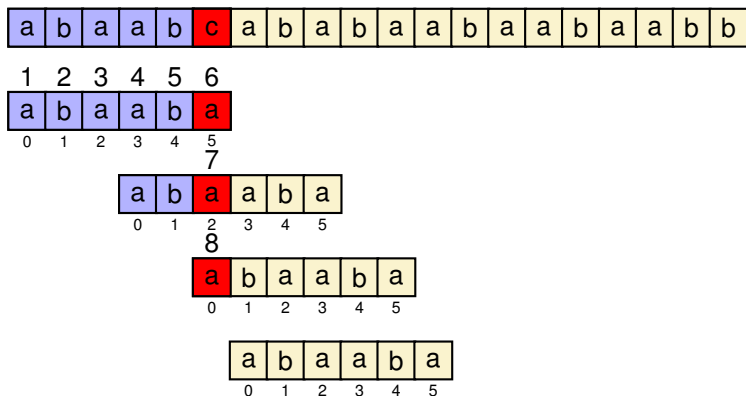
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

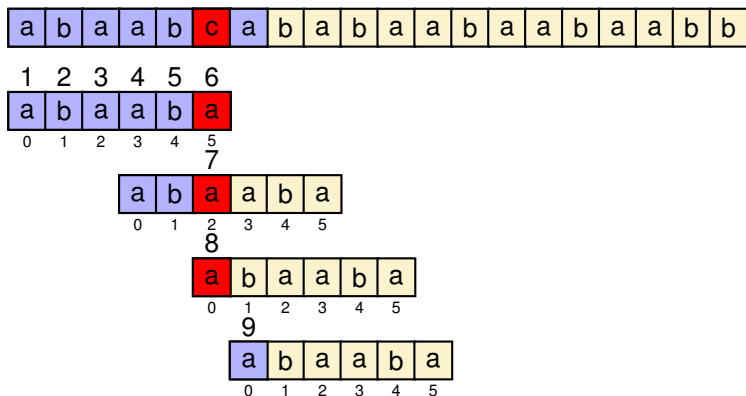
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

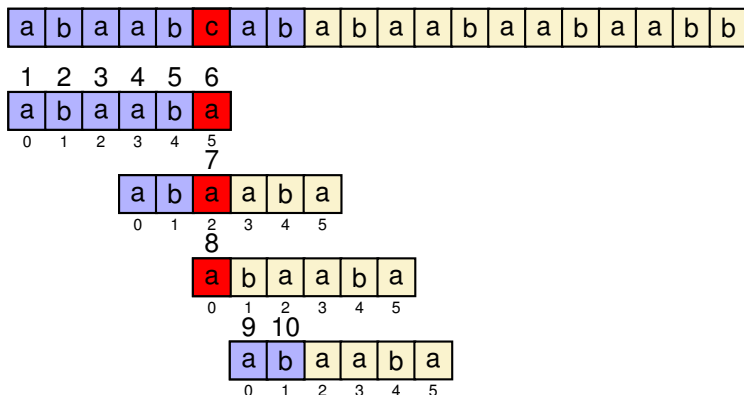
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

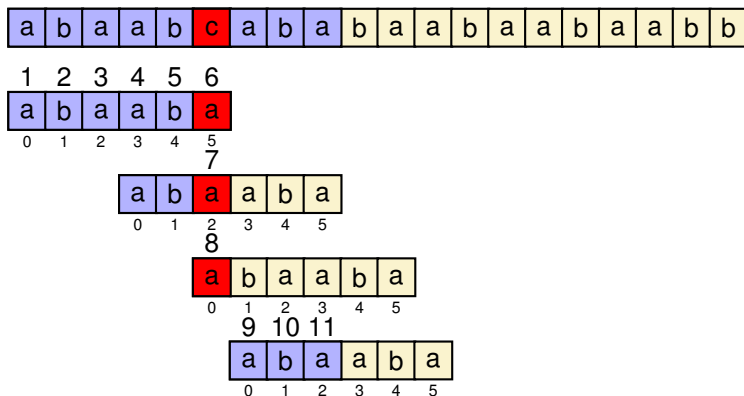
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

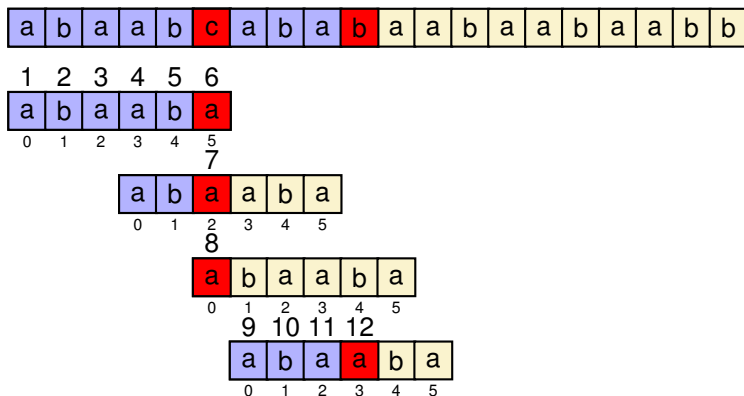
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

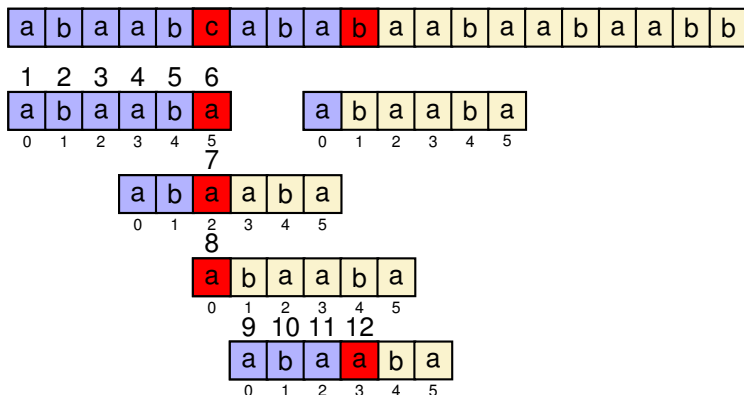
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

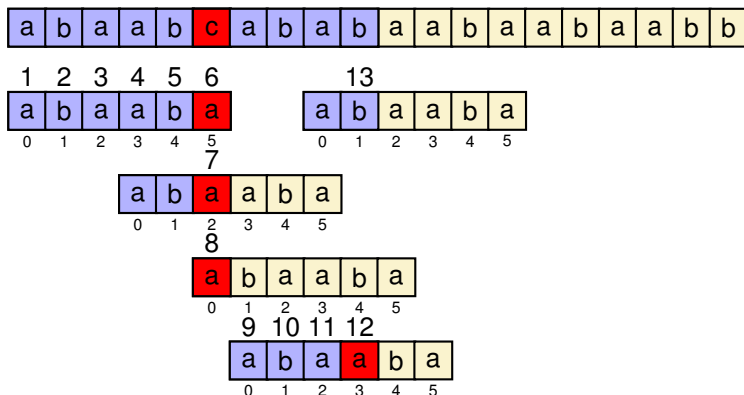
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

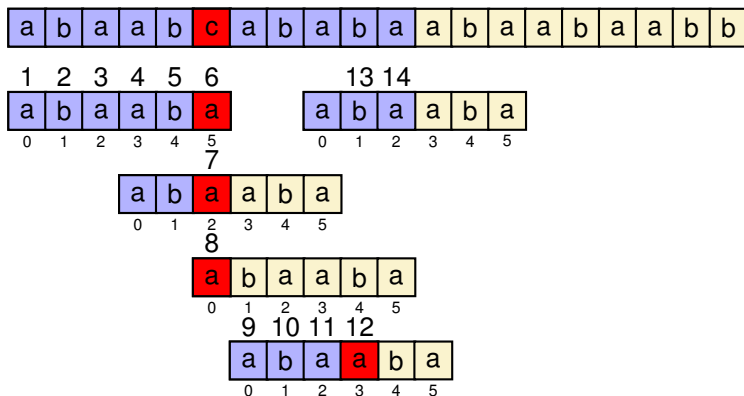
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

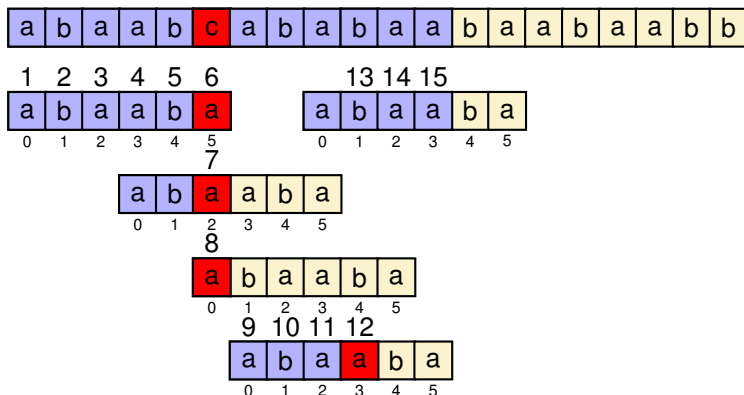
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

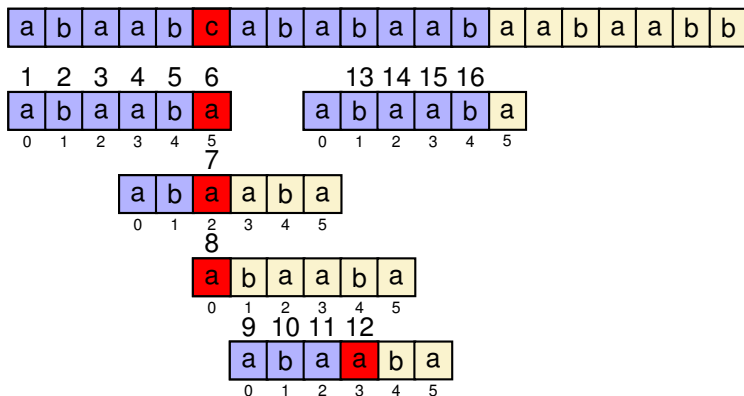
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

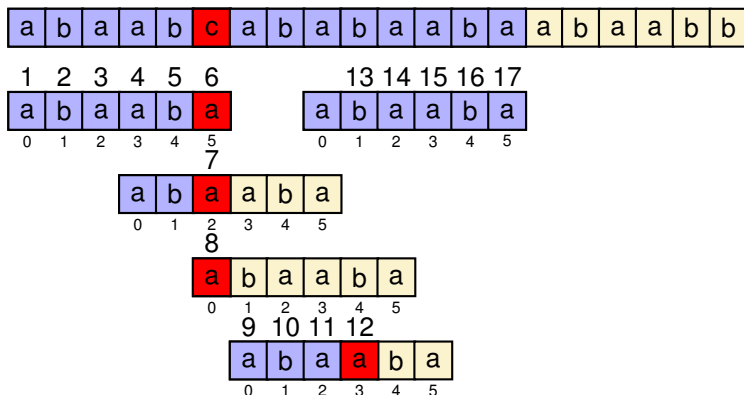
Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Example

Remember: when mismatch, we shift the pattern by $i - f(i - 1)$.



i	0	1	2	3	4	5
$f(i)$	0	0	1	1	2	3

Knuth-Morris-Pratt Algorithm

KMPMatch(T, P):

$F = \text{failureFunction}(P)$

$pos = 0$

$i = 0$

while $pos \leq n - m$ **do**

if $P[i] \neq T[pos + i]$ **then**

if $i > 0$ **then**

$pos = pos + i - f(i - 1)$

$i = f(i - 1)$

else

$pos = pos + 1$

else

$i = i + 1$

if $i == m$ **then return** pos (match at position pos)

done

return -1 (no match)

Knuth-Morris-Pratt: Analysis

- ▶ Failure function can be computed in $O(m)$.

Knuth-Morris-Pratt: Analysis

- ▶ Failure function can be computed in $O(m)$.
- ▶ In each iteration of the while-loop:
 - ▶ either i increases by 1, or
 - ▶ pos increases by at least the amount i decreases
- ▶ **While-loop is executed at most $2n$ times.**

Knuth-Morris-Pratt: Analysis

- ▶ Failure function can be computed in $O(m)$.
- ▶ In each iteration of the while-loop:
 - ▶ either i increases by 1, or
 - ▶ pos increases by at least the amount i decreases
- ▶ **While-loop is executed at most $2n$ times.**
- ▶ **Knuth-Morris-Pratt has time complexity $O(n + m)$.**

Overview

Pattern Matching

Boyer-Moore Algorithm

Knuth-Morris-Pratt Algorithm

Huffman code

Tries

Binary Character Encoding

Given: alphabet $\{A_1, \dots, A_n\}$.

Fixed-length code:

- ▶ every character get a code (a 0, 1 string) of fixed length
- ▶ examples: ASCII and Unicode-16
- ▶ example: $c(A) = 00$, $c(B) = 01$, $c(C) = 11$. Then

ACBA = 00110100

Binary Character Encoding

Given: alphabet $\{A_1, \dots, A_n\}$.

Fixed-length code:

- ▶ every character get a code (a 0, 1 string) of fixed length
- ▶ examples: ASCII and Unicode-16
- ▶ example: $c(A) = 00$, $c(B) = 01$, $c(C) = 11$. Then

ACBA = 00110100

Variable-length code:

- ▶ codes for each character may have different lengths
- ▶ example: **Huffman code**
- ▶ example: $c(A) = 0$, $c(B) = 10$, $c(C) = 11$. Then:

ACBA = 011100

Prefix-free codes

Example of an **ambiguous code**:

- ▶ $c(A) = 10$, for $c(B) = 01$, and for $c(C) = 0$. Then

$$BC = 010 \quad \text{and} \quad CA = 010$$

- ▶ we should avoid ambiguous codes

Prefix-free codes

Example of an **ambiguous code**:

- ▶ $c(A) = 10$, for $c(B) = 01$, and for $c(C) = 0$. Then

$$BC = 010 \quad \text{and} \quad CA = 010$$

- ▶ we should avoid ambiguous codes

A code is **prefix-free** if $c(A)$ is not prefix of $c(B)$ for all $A \neq B$

- ▶ prefix-free codes are non-ambiguous

Huffman code

Given:

- ▶ Alphabet A_1, A_2, \dots, A_n .
- ▶ Probabilities $0 \leq p(A_j) \leq 1$ for every letter A_j .

Problem:

- ▶ Find **variable length prefix-free code**.
- ▶ **Frequent characters should have shorter codes**.

Huffman code

Given:

- ▶ Alphabet A_1, A_2, \dots, A_n .
- ▶ Probabilities $0 \leq p(A_i) \leq 1$ for every letter A_i .

Problem:

- ▶ Find **variable length prefix-free code**.
- ▶ **Frequent characters should have shorter codes**.

Expected length for the codes, that is,

$$\sum_{1 \leq i \leq n} p(A_i) \cdot |c(A_i)|$$

should be minimal.

Huffman Algorithm: Idea

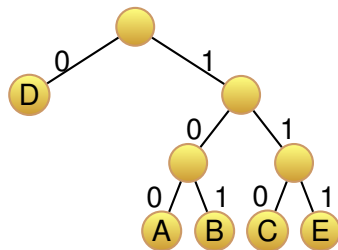
General idea:

- ▶ the algorithm constructs **binary trees**
- ▶ every leaf contains a character
- ▶ **path to the leaf is the code (0 for left, 1 for right)**

Huffman Algorithm: Idea

General idea:

- ▶ the algorithm constructs **binary trees**
- ▶ every leaf contains a character
- ▶ **path to the leaf is the code (0 for left, 1 for right)**



$$c(A) = 100$$

$$c(B) = 101$$

$$c(C) = 110$$

$$c(D) = 0$$

$$c(E) = 111$$

Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$

Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$

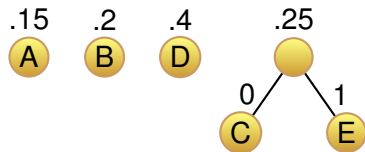
.15	.2	.15	.4	.1
				

Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$

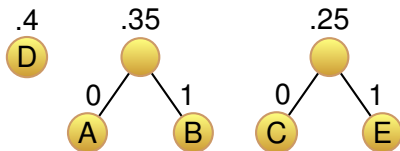


Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$

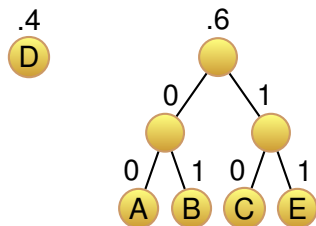


Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$

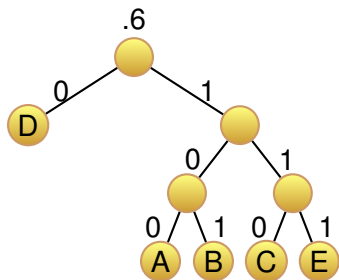


Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$



$$c(A) = 100$$

$$c(B) = 101$$

$$c(C) = 110$$

$$c(D) = 0$$

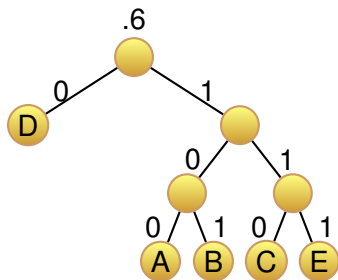
$$c(E) = 111$$

Huffman Algorithm

1. For every letter A create a tree consisting only of A .
2. Pick two trees with lowest sum of probabilities.
Merge these two trees with a new node at the root.
3. Repeat the last step until only one tree is left.

Example:

$$p(A) = .15, p(B) = .2, p(C) = .15, p(D) = .4, p(E) = .1$$



$$c(A) = 100$$

$$c(B) = 101$$

$$c(C) = 110$$

$$c(D) = 0$$

$$c(E) = 111$$

average code length:

$$3 \cdot 0.6 + 1 \cdot 0.4 = 2.2$$

Huffman Algorithm: Example

Example: $X = \text{abracadabra}$

Huffman Algorithm: Example

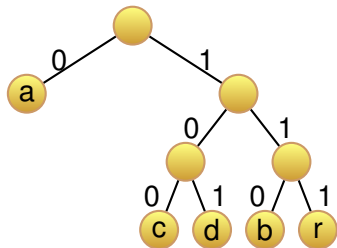
Example: $X = \text{abracadabra}$

a	b	c	d	r
5	2	1	1	2

Huffman Algorithm: Example

Example: $X = \text{abracadabra}$

a	b	c	d	r
5	2	1	1	2

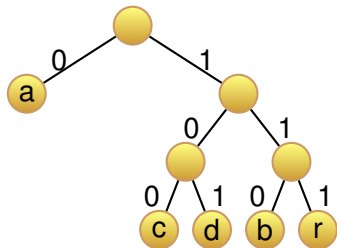


abracadabra =
01101110100010101101110
(length 23)

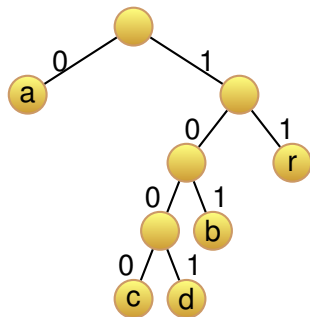
Huffman Algorithm: Example

Example: $X = \text{abracadabra}$

a	b	c	d	r
5	2	1	1	2



abracadabra =
01101110100010101101110
(length 23)



abracadabra =
01011101000010010101110
(length 23)

Huffman Algorithm: Analysis

This is a **greedy algorithm**:

- ▶ always pick the two trees with lowest key

Time complexity (to compute optimal code for a given word):

- ▶ $O(n + (d \log d))$
where n length of the word, d size of the alphabet

Extra Materiaal

- ▶ Boyer
- ▶ Moore
- ▶ Knuth
- ▶ Morris
- ▶ Pratt

Overview

Pattern Matching

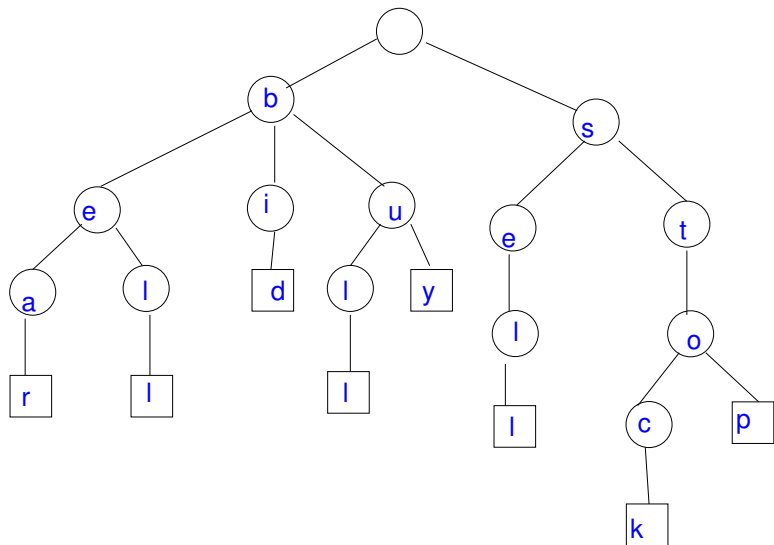
Boyer-Moore Algorithm

Knuth-Morris-Pratt Algorithm

Huffman code

Tries

Standard Trie: Example



contains strings: bear, bell, bid, bull, buy, sell, stock, stop

Standard Trie

- ▶ tree for storing strings
- ▶ no string is the prefix of another string
- ▶ empty root
- ▶ 1 character per node

Standard Trie: Properties

d size of the alphabet, s strings, n total length

- ▶ internal nodes have at most d children
- ▶ for every string there is precisely one leaf (external node)
- ▶ height = length of the longest string
- ▶ number of nodes $O(n)$

Standard Trie: Complexity

- ▶ **construction of a standard trie**
 - adding of a string of length m in $O(dm)$
 - construction of the whole trie $O(dn)$
- ▶ **search of a string of length m**
 - visits at most $m + 1$ nodes,
 - per node there are at most d checks
 - in $O(dm)$

Compressed Trie: Example

Replace path to leaf by leaf with a subword (suffix).

