

# Datastructuren

*Data Structures*

Hendrik Jan Hoogeboom

Informatica – LIACS  
Universiteit Leiden

najaar 2023

# Contents

- 10 Pattern Matching
  - Knuth-Morris-Pratt
  - Aho-Corasick
  - Comparing texts ☒

# patroonherkenning

## *woord (P) zoeken in tekst (T)*

veel algoritmen

- aantal letters    bv binair
- herhalingen in patroon
- preprocessing    meerdere malen gebruiken
  - KMP**    één voorwaartse scan
  - AC**    meerdere woorden tegelijk
- letters in P, vs letters in T, achterwaarts, hash

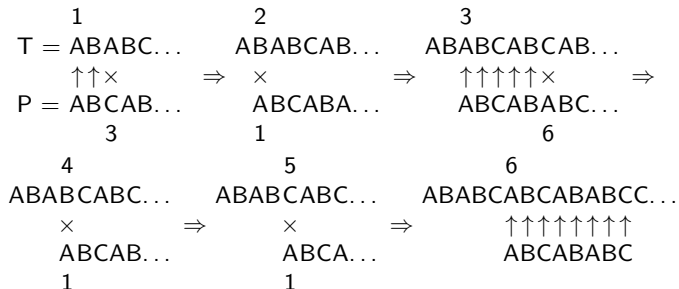
## *tekst (T) voldoet aan patroon (P)*

- reguliere expressie [opdracht 3]
  - eigenschap: palindroom
- wel/geen fouten
- bioinformatica**
- alignment [om te lezen]
  - gene prediction    signalen (geen harde karakterisatie)

## naive pattern matching

P = 'ABCABABC'

T = 'ABABCABCABABCC...'



# Contents

- 10 Pattern Matching
  - Knuth-Morris-Pratt
  - Aho-Corasick
  - Comparing texts ☒

# KMP

Donald Knuth, James H. Morris, and Vaughan Pratt (1970,1977)

- *failure links*
- linear time preprocessing pattern
- search will never back-up in text

match pattern against itself *niet zo!*

$$T = \dots \text{ABCABAB?} \dots$$

$$P = \quad \text{ABCABAB} \times$$

8

$$\begin{array}{cc} 2 & 8 \\ \text{ABCABAB} & . \end{array}$$

$$. \text{ABCABAB}$$

×

$$\begin{array}{cc} 3 & 8 \\ \text{ABCABAB} & . \end{array}$$

$$.. \text{ABCABAB}$$

×

$$\begin{array}{cc} 4 & 8 \\ \text{ABCABAB} & . \end{array}$$

$$... \text{ABCABAB}$$

×

$$\begin{array}{cc} 5 & 8 \\ \text{ABCABAB} & . \end{array}$$

$$.... \text{ABCABAB}$$

×

$$\begin{array}{cc} 6 & 8 \\ \text{ABCABAB} & . \end{array}$$

$$..... \text{ABCABAB}$$
3

## failure links

	2	3	4	5	6	7	8
A	AB	ABC	ABCA	ABCAB	ABCABA	ABCABAB	
.A	..AB	...ABC	... <u>AB</u> CA	... <u>AB</u> CAB	..... <u>AB</u> CA	..... <u>AB</u> CABAB	
1	1	1	<u>2</u>	<u>3</u>	<u>2</u>	<u>3</u>	

	k	1	2	3	4	5	6	7	8
P[k]	A	B	C	A	B	A	B	C	
FLink[k]	0	1	1	1	2	3	2	3	

at position  $k$ : the maximal  $r < k$  such that

$$P_1 \dots P_{r-1} = P_{k-r+1} \dots P_{k-1}$$

mismatch at position  $k$ , then continue at position  $\text{FLink}[k]$

(and *same* position in Text)

$\text{FLink}[k] = 0$ : next position in Text, first position Pattern



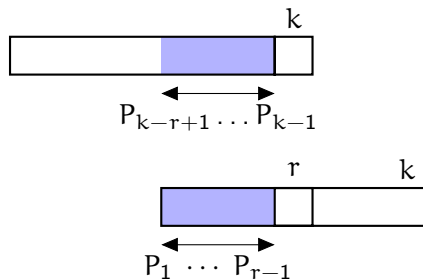
# Failure link

maximal suffix = prefix

$\text{FLink}[k] = r$

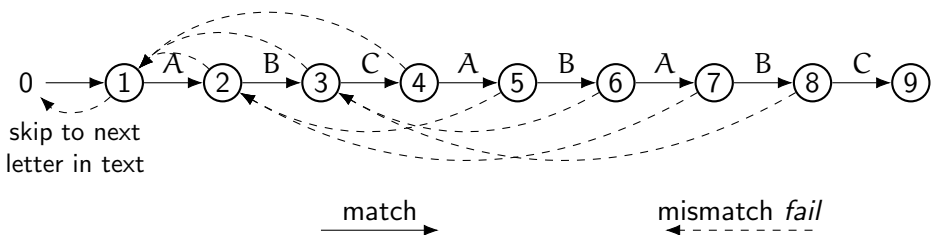
at position  $k$ : the maximal  $r < k$  such that

$$P_1 \dots P_{r-1} = P_{k-r+1} \dots P_{k-1}$$



## KMP failure links

k	1	2	3	4	5	6	7	8
P[k]	A	B	C	A	B	A	B	C
FLink[k]	0	1	1	1	2	3	2	3



# KMP search

## KMP using failure links

```
Pos = 1      // position in pattern
TPos = 1     // position in text
while ((Pos <= PatLen) and (TPos <= TextLen))
do if (P[Pos] == Text[TPos])
    then Pos ++;
        TPos ++;
    else Pos = FLink[Pos]
        if (Pos == 0) // next position in text
            then Pos = 1
                TPos ++;
        fi
    fi
od
if (Pos > PatLen) // found?
then Pattern found at position TPos-PatLen+1 in Text
fi
```

## using KMP links

k	1	2	3	4	5	6	7	8
P[k]	A	B	C	A	B	A	B	C
FLink[k]	0	1	1	1	2	3	2	3

1 3	3 8	6 8 13	
T = ABABC...	ABABCAB CAB...	BCABCABAB ABC...	
↑↑×	⇒ ↑↑↑↑↑×	⇒ ↑↑↑↑↑×	⇒
P = ABCAB...	ABCABABC...	ABCABABC	
3	1 6	3 8	

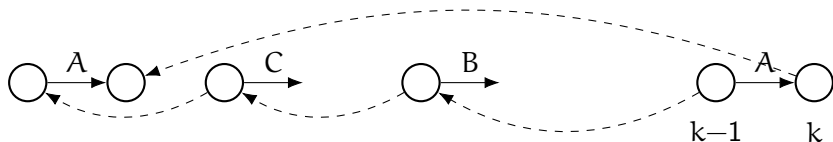
13	13	
ABAB ABC..	BAB ABC..	
×	⇒ ↑↑↑	⇒ ...
ABCABABC	ABCABABC	
3	1	



# KMP failure links

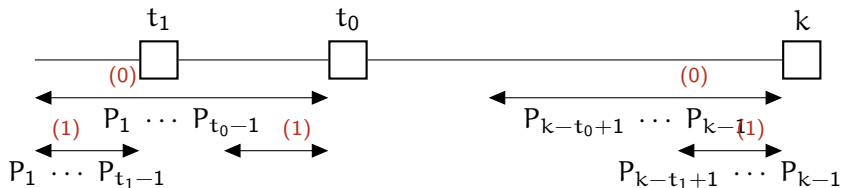
## computing KMP failure links

```
Pos = 1 // position in pattern
FLink[1] = 0
for Pos = 2 to PatLen
do Fail = FLink[Pos-1]
  while ( (Fail > 0) and (P[Fail] != P[Pos-1]) )
  do Fail = FLink[Fail]
  od
  FLink[Pos] = Fail+1
od
```



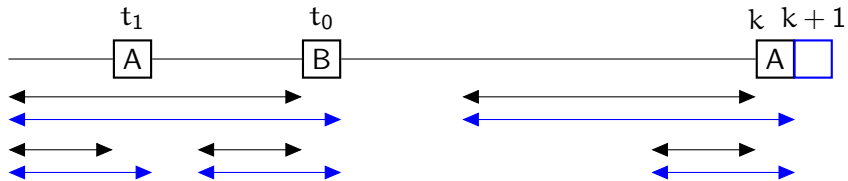
# why does this work

all prefixes that are also suffix  
can be found following failure links



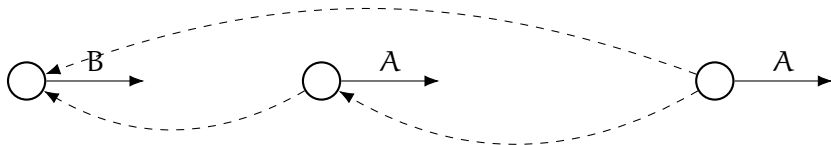
$$FLink[k] = t_0, \quad FLink[t_0] = t_1$$

# why ...





## link to same character



updating KMP failure links

```

for Pos = 2 to PatLen
do if ( P[Pos] == P[FLink[Pos]] )
    then FLink[Pos] = FLink[FLink[Pos]]
    fi
od

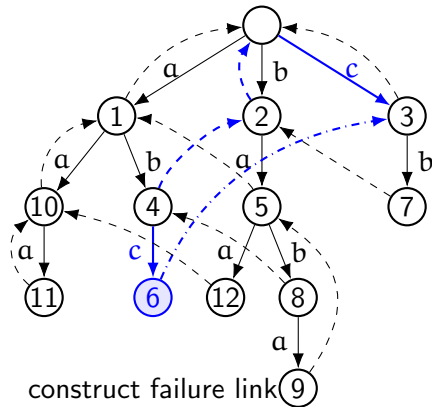
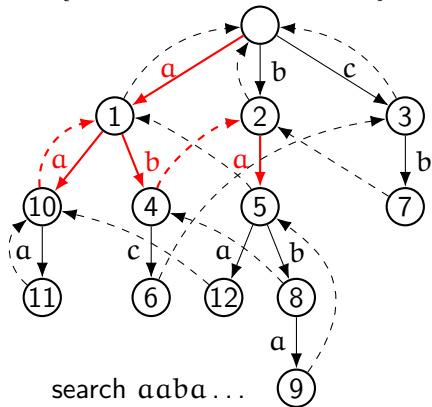
```

k	1	2	3	4	5	6	7	8
P[k]	A	B	C	A	B	A	B	C
FLink[k]	0	1	1	1	2	3	2	3
FLink'[k]	0	1	1	0	1	3	1	1

# Contents

- 10** Pattern Matching
  - Knuth-Morris-Pratt
  - Aho-Corasick
  - Comparing texts ☒

## Aho-Corasick

 $P = \{aaa, abc, baa, baba, cb\}$ 


# Contents

## 10 Pattern Matching

- Knuth-Morris-Pratt
- Aho-Corasick
- Comparing texts ☒

## alignment

## enzymes and their amino acids

```

82 TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
81 -----YECNQCQKAFQAQHSSLKCHYRTHIGEKPYECNQCQKAFSK 40
      ****: .***: * **:* * :****.:* *****..

```

```

82 PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHHTGEKPYE-CNQCQKAFQA- 116
81 HSHLQCHKRTHHTGEKPYECNQCQKAFSQHGLLQRHKRTHHTGEKPYMNVINMVKPLHNS 98
      **** *:*****:***:**.: .*****:          : *.: :

```

similarity TCAGACGATTG and TCGGAGCTG

TCAG-ACG-ATTG

TCAGACGATTG

TC-GGA-GC-T-G

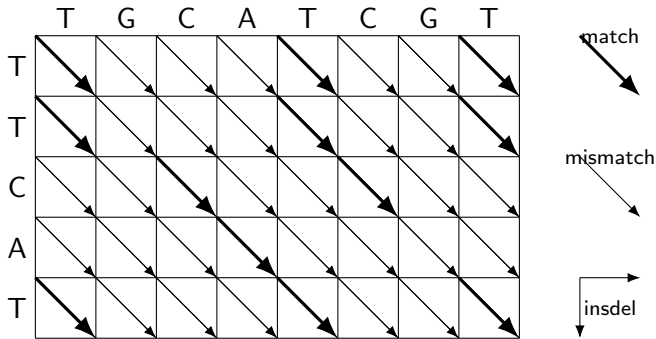
TCGGA-GCT-G

match, mismatch, insdel (gap)

G	A	-	A
G	G	G	-

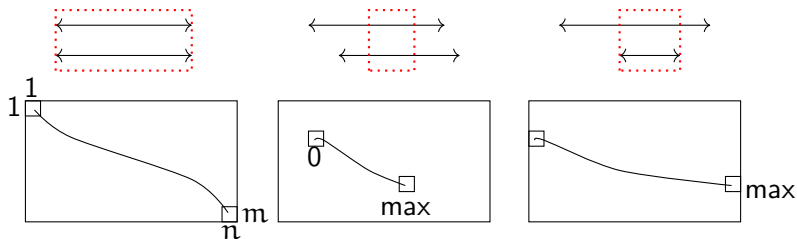
# global alignment

TTCAT vs. TGCATCGT



as shortest path

# global versus local alignment



Levenshtein distance (1966)

Needleman-Wunsch (1970)

Smith-Waterman (1981)



end.