

## Exaktes Pattern-Matching

Tutorium Algorithmen &  
Datenstrukturen für Bioinformatik  
Utz Pape; Adrian Haß

Boyer-Moore

1

## Boyer Moore Algorithmus

- Bad-Character-Rule (BC)
  - Beispiel
  - Vorverarbeitung
- Good-Suffix-Rule (GS)
  - Beispiele
  - Vorverarbeitung
- Anwendung beider Regeln

Boyer-Moore

2

## Ext. Bad Char. - Rule

- Gegeben:  
T Text; P Pattern:  $|T|=m; |P|=n$

Boyer-Moore

3

## Ext. Bad Char. - Rule

- Gegeben:  
T Text; P Pattern:  $|T|=m; |P|=n$
- Alignment:  $T[p\dots p+n] \sim P$

Boyer-Moore

4

## Ext. Bad Char. - Rule

- Gegeben:  
T Text; P Pattern:  $|T|=m; |P|=n$
- Alignment:  $T[p\dots p+n] \sim P$
- 1. Mismatch:  $T[k] \neq P[i]$

Boyer-Moore

5

## Ext. Bad Char

- Suche rechtestes Auftreten von  $T[k]$  in  $P$ , dass links von  $i$  liegt.

Boyer-Moore

6

## eBC-Rule

```
      0                1
      12345678901
T: ACgAgACAggT
      X |
P:  gACA
```

Boyer-Moore

7

## eBC-Rule

```
      0                1
      12345678901
T: ACgAgACAggT
      X |
P:  gACA
```

Boyer-Moore

8

## eBC-Rule

0	1
1	2
3	4
5	6
7	8
9	0
1	0

T: ACgAgACAggT

gACA			

P:

Boyer-Moore

9

## Pattern-Vorverarbeitung eBC-Rule

$\Sigma = \{A, C, g, T\}$  P=gACA

Liste mit Positionen( $\leftarrow$ ) in P:

- A:{4,2}
- C:{3}
- G:{1}
- T: {}

Boyer-Moore

10

## eBC-Rule

Abfrage:

Durchsuche Liste von T[k] nach Wert kleiner i!

Boyer-Moore

11

## Strong Goog Suffix - Rule

- Suche nächstes Auftreten  $P[i-1..n]$  (Suffix) in P, wobei der nächste Buchstabe  $\neq P[i]$  ist.

Boyer-Moore

12

## Strong Goog Suffix - Rule

- Suche nächstes Auftreten  $P[i-1..n]$  (Suffix) in  $P$ , wobei der nächste Buchstabe  $\neq P[i]$  ist.

Falls nicht vorhanden:

Boyer-Moore

13

## Strong Goog Suffix - Rule

- Suche nächstes Auftreten  $P[i-1..n]$  (Suffix) in  $P$ , wobei der nächste Buchstabe  $\neq P[i]$  ist.

Falls nicht vorhanden:

- Suche Prefix von  $P$  ( $P[1..i]$ ), der Suffix von  $P[n-l..n]$  entspricht.

Boyer-Moore

14

## sGS-Rule

1.Fall: Wir finden  $P[i..n]$  in  $P$  ganz

	0									1	
	1	2	3	4	5	6	7	8	9	0	
T:	g	T	A	C	g	A	T	g	A	C	T
					X						
P:	C	g	A	T	g	A					

Boyer-Moore

15

## sGS-Rule

1.Fall: Wir finden  $P[i..n]$  in  $P$  ganz

	0									1	
	1	2	3	4	5	6	7	8	9	0	
T:	g	T	A	C	g	A	T	g	A	C	T
P:	C	g	A	T	g	A					

Boyer-Moore

16

## sGS-Rule

2.Fall: Suchen Prefix(P) = Suffix(P)

0                      1  
123456789012  
T: gTACTgATTgAC  
    X| | |  
P: gATTgA

Boyer-Moore

17

## sGS-Rule

2.Fall: Suchen Prefix(P) = Suffix(P)

0                      1  
123456789012  
T: gTACTgATTgAC  
          | | | | | |  
P: gATTgA

Boyer-Moore

18

## Pattern-Vorverarbeitung sGS-Rule

Beobachtung:

- $P[i..n]$  soll gleich  $P[k..(k+n-i)]$  sein
- Dreht man P um, so erhält man
- $P[1..(n-i)]$  soll gleich  $P[l..(l+n-i)]$  sein

→ Z-Algorithmus

Boyer-Moore

19

## Anwendung beider Regeln

- Nehmen immer  $\max\{eBC, sGS, 1\}$

Boyer-Moore

20

## Beide Regeln

P: CgACCg

T: ACAgACTgTCACggACCgACCg

- Sprungliste P:  
A:{3};g:{6,2};C:{5,4,1};T: {}
- Prefix-Suffix:  
geschultes Auge...

Boyer-Moore

21

## Beide Regeln

T: ACAgACTgTCACggACCgACCg

X

P: CgACCg

Boyer-Moore

22

## Beide Regeln

T: ACAgACTgTCACggACCgACCg

X

P: CgACCg

GS:1

BC:1

→1

Boyer-Moore

23

## Beide Regeln

T: ACAgACTgTCACggACCgACCg

X

P: CgACCg

Boyer-Moore

24

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

X

P: CgACCg

GS:1

BC:6 (T kommt nicht vor)

→6

Boyer-Moore

25

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

X | |

P: CgACCg

Boyer-Moore

26

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

X | |

P: CgACCg

GS:4

BC:1

→4

Boyer-Moore

27

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

| | X

P: CgACCg

Boyer-Moore

28

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

|| X

P: CgACCg

GS:1

BC:1

→1

Boyer-Moore

29

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

X||||

P: CgACCg

Boyer-Moore

30

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

X||||

P: CgACCg

GS:4

BC:1

→4

Boyer-Moore

31

## Beide Regeln

T: ACAGACTgTCACggACCgACCg

||||

P: CgACCg

MATCH!

Boyer-Moore

32



ENDE