

# Algorithmen auf Sequenzen

## Exakte Mustersuche: Sublineare Algorithmen

Sven Rahmann

Genominformatik  
Universitätsklinikum Essen  
Universität Duisburg-Essen  
Universitätsallianz Ruhr

# Sublineare Algorithmen zur exakten Mustersuche



- Bisherige Algorithmen haben jeden Buchstaben im Text exakt einmal gelesen.
- Kleinstmögliche Anzahl von Vergleichen zwischen  $P$  und  $T$ ?

# Sublineare Algorithmen zur exakten Mustersuche

- Bisherige Algorithmen haben jeden Buchstaben im Text exakt einmal gelesen.
- Kleinstmögliche Anzahl von Vergleichen zwischen  $P$  und  $T$ ?
- Wenn das Muster von rechts verglichen wird und der Textbuchstabe nicht in  $P$  vorkommt, kann man das Muster sofort um  $m$  Stellen verschieben.
- Wenn aber in einem Block von  $m$  Textzeichen keins vergleicht, kann man das Muster übersehen.
- Eine untere Schranke für die exakte Mustersuche ist demnach  $\Omega(n/m)$ .

# Szenario für untere Schranke

$$T = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline & & & a & & & & a & & & & a & & & & & & \\ \hline \end{array}$$
$$P = \begin{array}{|c|c|c|c|} \hline b & b & b & b \\ \hline \end{array}$$

Das Muster kann immer um  $m$  Stellen verschoben werden.

# Szenario für untere Schranke

$$T = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline & & & a & & & & a & & & & a & & & & \\ \hline \end{array}$$
$$P = \begin{array}{|c|c|c|c|} \hline b & b & b & b \\ \hline \end{array}$$

Das Muster kann immer um  $m$  Stellen verschoben werden.

Allgemein: Für jedes  $\sigma \in \Sigma$  ermitteln, wie weit verschoben werden darf, ohne eine Instanz des Musters zu übersehen.

Der **Horspool-Algorithmus** realisiert diesen Ansatz.

Beginnend mit einem Suchfenster ganz links im Text, bis das Suchfenster rechts den Text verlässt:

- Das rechteste Zeichen  $c$  des Text-Suchfensters wird mit  $P[m - 1]$  verglichen.
- Berechne Verschiebung an Hand von  $c$ .
- Bei Nichtübereinstimmung: Verschiebe sofort.
- Bei Übereinstimmung: Vergleiche  $P$  mit Textabschnitt, verschiebe dann.

Berechnung der Verschiebung (shift): Sei

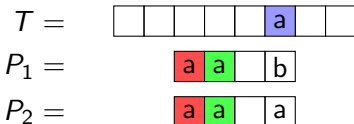
$$\ell[c] := \max\{\{0 \leq j < m - 1 : P[j] = c\} \cup \{-1\}\}$$

die rechteste Position von  $c$  in  $P[: m - 1]$  (bzw.  $-1$ )

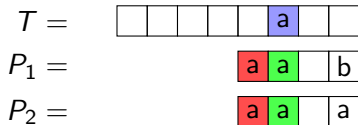
Verschiebung:  $shift[c] := m - 1 - \ell[c]$

# Der Horspool-Algorithmus

Vor der Verschiebung:



Nach der Verschiebung:



Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    7

b    7

c    7



Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    7

b    7

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     **a** b a b a c a

*shifts* =

a    7

b    7

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    6

b    7

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

$shifts =$

a    6

b    7

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    6

b    5

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    6

b    5

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    4

b    5

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a **b** a c a

*shifts* =

a    4

b    5

c    7



Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    4

b    3

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    4

b    3

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    2

b    3

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    2

b    3

c    7

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    2

b    3

c    1

Erstellen der Verschiebungstabelle:

```
1 def create_shifts(P, S):  
2     m = len(P)  
3     shifts = {s: m for s in S}  
4     for i, c in enumerate(P[:m-1]):  
5         shifts[c] = m - 1 - i  
6     return shifts
```

$P =$     a b a b a c a

*shifts* =

a    2

b    3

c    1

```
1 def horspool(P, T):
2     m, n, S = len(P), len(T), set(P + T)
3     shifts = create_shifts(P, S)
4     lastP = P[-1]
5     last = m-1 # rechteste Pos. des Suchfensters
6     while True:
7         while last < n and T[last] != last_P:
8             last += shifts[T[last]]
9         if last >= n:
10            break
11        if T[last-m+1:last] == P[:m-1]:
12            yield (last-m+1, last+1)
13        last += shifts[last_P]
```

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 6$

$shifts:$

a: 2, b: 3, c: 1



Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 6$

$shifts:$

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 6$

$shifts:$

$a: 2, b: 3, c: 1$

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 9$

$shifts:$

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 9$

$shifts:$

a: 2, b: 3, c: 1

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 12$

*shifts:*

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 12$

$shifts:$

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 12$

$shifts:$

$a: 2, b: 3, c: 1$

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 12$

*shifts:*

a: 2, b: 3, c: 1



Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 14$

$shifts:$

$a: 2, b: 3, c: 1$

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 14$

$shifts:$

$a: 2, b: 3, c: 1$

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 14$

$shifts:$

$a: 2, b: 3, c: 1$

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 14$

$shifts:$

yield 8,15

a: 2, b: 3, c: 1

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 14$

$shifts:$

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 16$

$shifts:$

$a: 2, b: 3, c: 1$

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 16$

$shifts:$

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 17$

$shifts:$

a: 2, b: 3, c: 1



Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 17$

$shifts:$

a: 2, b: 3, c: 1

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 18$

$shifts:$

$a: 2, b: 3, c: 1$

# Der Horspool-Algorithmus

Mustersuche mit dem Horspool-Algorithmus:

```
1 while True:
2     while last < n and T[last] != last_P:
3         last += shifts[T[last]]
4     if last >= n: break
5     if T[last-m+1:last] == P[:m-1]:
6         yield (last-m+1, last+1)
7     last += shifts[last_P]
```

$T =$ 

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

$P =$ 

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

$last = 18$

$shifts:$

$a: 2, b: 3, c: 1$

- Best-case-Laufzeit:  $\Omega(m + n/m)$
- Worst-case-Laufzeit:  $\mathcal{O}(nm)$ .

- Best-case-Laufzeit:  $\Omega(m + n/m)$
- Worst-case-Laufzeit:  $\mathcal{O}(nm)$ .
- Der Horspool-Algorithmus ist eine vereinfachte Version des *Boyer-Moore-Algorithmus*, welcher auch eine Best-case-Laufzeit von  $\Omega(m + n/m)$ , aber eine Worst-case-Laufzeit von  $\mathcal{O}(m + n)$  hat.

- Best-case-Laufzeit:  $\Omega(m + n/m)$
- Worst-case-Laufzeit:  $\mathcal{O}(nm)$ .
- Der Horspool-Algorithmus ist eine vereinfachte Version des *Boyer-Moore-Algorithmus*, welcher auch eine Best-case-Laufzeit von  $\Omega(m + n/m)$ , aber eine Worst-case-Laufzeit von  $\mathcal{O}(m + n)$  hat.
- Erwartete Laufzeit: Erwartete Verschiebungslänge in vielen Szenarien  $\Theta(m)$ , erwartete Laufzeit für Suche dann  $\mathcal{O}(n/m)$ .  
Nicht für "kleine Alphabete"

- Sei  $\Sigma_P$  die Menge der Zeichen in  $P$
- Zeichen in  $\Sigma \setminus \Sigma_P$  haben Shiftlänge  $m$ .
- Worst case: Die Zeichen in  $\Sigma_P$  haben kleinstmögliche Shiftlängen  $1, \dots, |\Sigma_P|$ .
- Annahme: Text ist zufällig i.i.d.; d.h. letztes Zeichen jedes Suchfensters ist zufällig.

# Laufzeitanalyse

- Sei  $\Sigma_P$  die Menge der Zeichen in  $P$
- Zeichen in  $\Sigma \setminus \Sigma_P$  haben Shiftlänge  $m$ .
- Worst case: Die Zeichen in  $\Sigma_P$  haben kleinstmögliche Shiftlängen  $1, \dots, |\Sigma_P|$ .
- Annahme: Text ist zufällig i.i.d.; d.h. letztes Zeichen jedes Suchfensters ist zufällig.

Erwartete Shiftlänge  $s$  erfüllt:

$$s \geq \frac{\sum_{i=1}^{|\Sigma_P|} i + (|\Sigma| - |\Sigma_P|) \cdot m}{|\Sigma|} = \frac{|\Sigma_P|(|\Sigma_P| + 1)}{2|\Sigma|} + m \left( 1 - \frac{|\Sigma_P|}{|\Sigma|} \right).$$



# Laufzeitanalyse

- Sei  $\Sigma_P$  die Menge der Zeichen in  $P$
- Zeichen in  $\Sigma \setminus \Sigma_P$  haben Shiftlänge  $m$ .
- Worst case: Die Zeichen in  $\Sigma_P$  haben kleinstmögliche Shiftlängen  $1, \dots, |\Sigma_P|$ .
- Annahme: Text ist zufällig i.i.d.; d.h. letztes Zeichen jedes Suchfensters ist zufällig.

Erwartete Shiftlänge  $s$  erfüllt:

$$s \geq \frac{\sum_{i=1}^{|\Sigma_P|} i + (|\Sigma| - |\Sigma_P|) \cdot m}{|\Sigma|} = \frac{|\Sigma_P|(|\Sigma_P| + 1)}{2|\Sigma|} + m \left( 1 - \frac{|\Sigma_P|}{|\Sigma|} \right).$$

Also  $s \in \Theta(m)$ , sobald  $|\Sigma| > |\Sigma_P|$  oder  $\Sigma_P \in \Theta(m)$ .

Fall  $|\Sigma| = |\Sigma_P| \in O(1)$  liefert konstantes  $\tilde{s}$ .

- Verschiebungen des Suchfensters um bis zu  $m$  Zeichen; dadurch Laufzeit  $\Theta(m + n/m)$  möglich
- Horspool-Algorithmus ist einfach und daher schnell (bei großem Alphabet und/oder  $|\Sigma_P| \ll |\Sigma|$ )
- Verschiebungen nicht optimal
- Idee: Bessere Verschiebungen mit mehr Informationen?

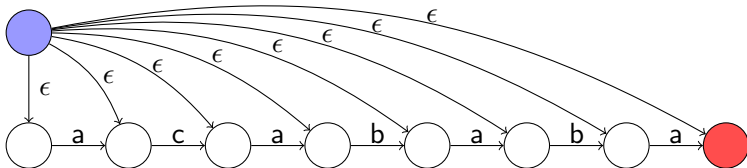
- Mehr Informationen pro Suchfenster sammeln:  
Längstes Suffix des Suchfensters, das auch Teilstring / Präfix des Musters ist?
- Eine Datenstruktur hierfür muss erlauben:
  - einem gelesenen Fenster von rechts nach links Zeichen anzufügen,
  - festzustellen, ob der bisher gelesene Teil ein Teilstring des Patterns ist,
  - festzustellen, ob der bisher gelesene Teil sogar ein Präfix des Patterns ist.

Der **Suffixautomat** für den String  $x$  ist ein Automat mit folgenden Eigenschaften:

- Es existiert vom Startzustand aus ein Pfad mit Label  $y$  genau dann, wenn  $y$  ein Teilstring von  $x$  ist.
- Der Pfad mit Label  $y$  endet genau dann in einem akzeptierenden Zustand, wenn  $y$  ein Suffix von  $x$  ist.
- Es muss nicht zu jedem Zustand und jedem Buchstaben eine ausgehende Kante geben.

Wird der Suffixautomat für das reverse Pattern  $P^{\text{rev}}$  zu  $P$  konstruiert, so erlaubt die zweite Eigenschaft das Erkennen von Präfixen von  $P$ .

$P = ababaca$



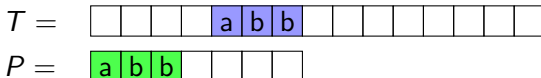
Suffixautomat für das reverse Pattern  $P^{\text{rev}} = acababa$ .

Eigenschaften:

- Zu Beginn alle Zustände aktiv.
- Aktive Zustände, solange gelesene Zeichenfolge Teilstring des Musters ist.
- Akzeptiert, wenn Suffix des Musters gelesen wurde.

# Backward Nondeterministic DAWG Matching (BNDM)

- Suffixautomat ist DAWG: *directed acyclic word graph*
- Suffixautomat wird zu  $P^{\text{rev}}$  konstruiert
- Bitparallele Realisierung, ähnlich Shift-And (für  $|P| \leq W$ )  
Aktive Zustandsmenge wird durch Verschieben und *Verunden* mit Masken aktualisiert
- Präfix  $P[:j]$  gefunden, wenn akzeptierend nach  $j$  gelesenen Zeichen



- Ist  $j = m$ , dann wird ein Match gemeldet.
- Ist  $0 \leq j < m$ , wird  $j$  als *lastsuffix* gespeichert.
- Das Suchfenster wird um  $m - \textit{lastsuffix}$  Positionen verschoben.

```
1 def bndm(P, T):
2     m, n = len(P), len(T)
3     full = (1 << m) - 1 # m ones
4     accept = 1 << (m - 1)
5     masks = create_masks(P[::-1], set(P+T))
6     last = m # Position direkt hinter Suchfenster
7     ...
```

# Schleife des BNDM-Algorithmus

```
1      ...
2      while last <= n:
3          A = full
4          lastsuffix = 0
5          j = 1
6          while A:
7              A &= masks[T[last-j]]
8              if A & accept:
9                  if j == m:
10                     yield (last-m, last)
11                     break
12                 else:
13                     lastsuffix = j
14             j += 1
15             A <<= 1
16         last += m-lastsuffix
```



# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 7$      $j = -$      $A = -$

*mask:* a: 1010101  
b: 0101000  
c: 0000010

$lastsuffix = -$

# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 7$      $j = -$      $A = -$

*mask:* a: 1010101  
b: 0101000  
c: 0000010

$lastsuffix = -$

# BNDM: Beispiel

$T =$	<table border="1"><tr><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>a</td><td>b</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td></tr></table>	a	b	c	a	b	a	b	a	c	a	b	c	$mask:$	a: 1010101
a	b	c	a	b	a	b	a	c	a	b	c				
$P =$	<table border="1"><tr><td>a</td><td>b</td><td>a</td><td>b</td><td>a</td><td>c</td><td>a</td></tr></table>	a	b	a	b	a	c	a		b: 0101000					
a	b	a	b	a	c	a									
			c: 0000010												
$last = 7$	$j = 1$	$A = 1111111$	$lastsuffix = 0$												

# BNDM: Beispiel

$T =$	<table border="1"><tr><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>a</td><td>b</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td></tr></table>	a	b	c	a	b	a	b	a	c	a	b	c	$mask:$	a: 1010101
a	b	c	a	b	a	b	a	c	a	b	c				
$P =$	<table border="1"><tr><td>a</td><td>b</td><td>a</td><td>b</td><td>a</td><td>c</td><td>a</td></tr></table>	a	b	a	b	a	c	a		b: 0101000					
a	b	a	b	a	c	a									
			c: 0000010												
$last = 7$	$j = 1$	$A = 1111111$	$lastsuffix = 0$												

# BNDM: Beispiel

$T =$ 

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

 $mask:$  a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 1$        $A = 0101000$        $lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 1$        $A = 0101000$        $lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 7$        $j = 2$        $A = 1010000$        $lastsuffix = 0$

# BNDM: Beispiel

$T =$	<table border="1"><tr><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>a</td><td>b</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td></tr></table>	a	b	c	a	b	a	b	a	c	a	b	c	$mask:$	a: 1010101
a	b	c	a	b	a	b	a	c	a	b	c				
			b: 0101000												
			c: 0000010												
$P =$	<table border="1"><tr><td>a</td><td>b</td><td>a</td><td>b</td><td>a</td><td>c</td><td>a</td></tr></table>	a	b	a	b	a	c	a							
a	b	a	b	a	c	a									
$last = 7$	$j = 2$	$A = 1010000$	$lastsuffix = 0$												



# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 7$       $j = 2$       $A = 1010000$       $lastsuffix = 0$

$mask:$  a: 1010101  
b: 0101000  
c: 0000010

# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 7$       $j = 2$       $A = 1010000$       $lastsuffix = 0$

*mask:* a: 1010101  
b: 0101000  
c: 0000010

# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 7$      $j = 2$      $A = 1010000$

*mask:* a: 1010101  
b: 0101000  
c: 0000010

$lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 7$        $j = 2$        $A = 1010000$        $lastsuffix = 2$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 3$        $A = 0100000$        $lastsuffix = 2$

# BNDM: Beispiel

$T =$ 

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

 $mask:$  a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 3$        $A = 0100000$        $lastsuffix = 2$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 7$        $j = 3$        $A = 0100000$        $lastsuffix = 2$

# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 7$      $j = 3$      $A = 0100000$      $lastsuffix = 2$

$mask:$     a: 1010101  
            b: 0101000  
            c: 0000010



# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 7$        $j = 4$        $A = 1000000$        $lastsuffix = 2$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 4$        $A = 1000000$        $lastsuffix = 2$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

b: 0101000  
c: 0000010

$last = 7$       $j = 4$       $A = 1000000$       $lastsuffix = 2$

$T =$ 

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

*mask:* a: 1010101

$P =$ 

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

      b: 0101000

$last = 7$      $j = 4$      $A = 1000000$      $lastsuffix = 2$       c: 0000010

$T =$ 

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

$P =$ 

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

$last = 7$       $j = 4$       $A = 1000000$

$mask:$    a: 1010101  
           b: 0101000  
           c: 0000010

$lastsuffix = 2$

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 4$        $A = 1000000$        $lastsuffix = 4$

# BNDM: Beispiel

$T =$ 

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

 $mask:$  a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 7$        $j = 5$        $A = 0000000$        $lastsuffix = 4$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101

$P =$ 

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

      b: 0101000

$last = 7$        $j = 5$        $A = 0000000$        $lastsuffix = 4$       c: 0000010



# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 5$      $A = 0000000$      $lastsuffix = 4$

# BNDM: Beispiel

$T =$ 

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

*mask:*  a: 1010101  
 $P =$ 

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

           b: 0101000  
 $last = 10$       $j = 5$        $A = 0000000$      $lastsuffix = 4$       c: 0000010

# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 10$      $j = 1$      $A = 1111111$

*mask*: a: 1010101  
b: 0101000  
c: 0000010

$lastsuffix = 0$

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 1$      $A = 1111111$      $lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 1$      $A = 1010101$      $lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 1$      $A = 1010101$      $lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 1$      $A = 1010101$      $lastsuffix = 0$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 1$      $A = 1010101$      $lastsuffix = 1$



# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 2$      $A = 0101010$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101

$P =$ 

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

     b: 0101000

$last = 10$       $j = 2$       $A = 0101010$      c: 0000010

$lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 2$      $A = 0000010$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

$P =$ 

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

$last = 10$       $j = 2$       $A = 0000010$

*mask:* a: 1010101

b: 0101000

c: 0000010

$lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 3$      $A = 0000100$      $lastsuffix = 1$

# BNDM: Beispiel

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

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

*mask:* a: 1010101  
b: 0101000  
c: 0000010

$last = 10$      $j = 3$      $A = 0000100$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$	a	b	c	a	b	a	b	a	c	a	b	c
-------	---	---	---	---	---	---	---	---	---	---	---	---

$P =$	a	b	a	b	a	c	a
-------	---	---	---	---	---	---	---

$last = 10$     $j = 3$     $A = 0000100$     $mask:$  a: 1010101  
b: 0101000  
c: 0000010

$lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

 $mask:$  a: 1010101

$P =$ 

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

      b: 0101000

$last = 10$      $j = 3$      $A = 0000100$      $lastsuffix = 1$       c: 0000010



# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 4$      $A = 0001000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
b: 0101000  
c: 0000010

$P =$ 

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

$last = 10$      $j = 4$      $A = 0001000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 4$      $A = 0001000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 4$      $A = 0001000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

 $mask:$ 

a:	1	0	1	0	1	0	1
b:	0	1	0	1	0	0	0
c:	0	0	0	0	0	1	0

$P =$ 

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

$last = 10$      $j = 5$      $A = 0010000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 5$      $A = 0010000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 5$      $A = 0010000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 5$      $A = 0010000$      $lastsuffix = 1$



# BNDM: Beispiel

$T =$ 

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

 $mask:$  a: 1010101  
          b: 0101000  
          c: 0000010

$P =$ 

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

$last = 10$      $j = 6$      $A = 0100000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
b: 0101000  
c: 0000010

$P =$ 

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

$last = 10$      $j = 6$      $A = 0100000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010  
 $last = 10$      $j = 6$      $A = 0100000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101

$P =$ 

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

      b: 0101000

$last = 10$      $j = 6$      $A = 0100000$      $lastsuffix = 1$       c: 0000010

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 10$      $j = 7$      $A = 1000000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

 $mask:$     a: 1010101  
 $P =$ 

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

                   b: 0101000  
                       c: 0000010

$last = 10$      $j = 7$      $A = 1000000$      $lastsuffix = 1$



$T =$ 

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

$P =$ 

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

$last = 10$

$j = 7$

$A = 1000000$

$mask:$  a: 1010101

b: 0101000

c: 0000010

$lastsuffix = 1$



# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      yield 3, 10      b: 0101000  
 $last = 10$      $j = 7$      $A = 1000000$      $lastsuffix = 1$       c: 0000010

# BNDM: Beispiel

$T =$ 

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

*mask:*   a: 1010101

$P =$ 

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

                b: 0101000

$last = 10$      $j = 7$      $A = 1000000$       c: 0000010

$lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

      b: 0101000  
c: 0000010

$last = 16$      $j = 7$      $A = 1000000$      $lastsuffix = 1$

# BNDM: Beispiel

$T =$ 

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

*mask:* a: 1010101  
 $P =$ 

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

       b: 0101000  
       c: 0000010  
 $last = 16$      $j = 7$      $A = 1000000$      $lastsuffix = 1$

- BNDM nutzt nichtdeterministischen Suffixautomaten zu  $P^{\text{rev}}$ , um Präfixe des Musters im Suchfenster zu finden.
- BNDM nutzt Bitparallelismus zur Simulation des Suffixautomaten ( $|P| < W$ )
- Deterministisch z.B. mit Suffixbaum zu  $P^{\text{rev}}$  (später)
- Längere Shifts als Horspool vs. mehr Aufwand pro Fenster
- Best-case Laufzeit  $\mathcal{O}(m + n/m)$ ,
- Worst-case Laufzeit  $\mathcal{O}(m + nm)$

<i>Algorithmus</i>	<i>Best-case</i>	<i>Worst-case</i>	<i>Speicher</i>
Naiv	$\Omega(n)$	$\mathcal{O}(mn)$	$\mathcal{O}(1)$
KMP	$\Omega(m + n)$	$\mathcal{O}(m + n)$	$\mathcal{O}(m)$
Shift-And*	$\Omega(m + n)$	$\mathcal{O}(m + n)$	$\mathcal{O}( \Sigma )$
Horspool	$\Omega(m + n/m)$	$\mathcal{O}(mn)$	$\mathcal{O}(m +  \Sigma )$
BNDM*	$\Omega(m + n/m)$	$\mathcal{O}(mn)$	$\mathcal{O}(m +  \Sigma )$

\*: bit-parallel,  $|P| < \text{Registerbreite}$

<i>Algorithmus</i>	<i>Best-case</i>	<i>Worst-case</i>	<i>Speicher</i>
Naiv	$\Omega(n)$	$\mathcal{O}(mn)$	$\mathcal{O}(1)$
KMP	$\Omega(m + n)$	$\mathcal{O}(m + n)$	$\mathcal{O}(m)$
Shift-And*	$\Omega(m + n)$	$\mathcal{O}(m + n)$	$\mathcal{O}( \Sigma )$
Horspool	$\Omega(m + n/m)$	$\mathcal{O}(mn)$	$\mathcal{O}(m +  \Sigma )$
BNDM*	$\Omega(m + n/m)$	$\mathcal{O}(mn)$	$\mathcal{O}(m +  \Sigma )$

\*: bit-parallel,  $|P| < \text{Registerbreite}$

Diskussion: Welcher Algorithmus für welches Szenario?