

# Algorithmen auf Sequenzen

## Exakte Mustersuche: DFAs, KMP-Algorithmus

Sven Rahmann

Genominformatik  
Universitätsklinikum Essen  
Universität Duisburg-Essen  
Universitätsallianz Ruhr

Problem des NFA: Verwaltung von  $\mathcal{O}(m)$  Zuständen pro Schritt  
Idee: Äquivalenten DFA konstruieren (nur ein aktiver Zustand)

Problem des NFA: Verwaltung von  $\mathcal{O}(m)$  Zuständen pro Schritt  
Idee: Äquivalenten DFA konstruieren (nur ein aktiver Zustand)

## Definition

Ein deterministischer endlicher Automat (DFA) ist ein Tupel  $(Q, q_0, F, \Sigma, \delta)$ , wobei

- $Q$  eine endliche Menge von Zuständen,
- Startzustand  $q_0 \in Q$ ,
- $F \subset Q$  eine Menge von akzeptierenden Zuständen,
- $\Sigma$  das Eingabealphabet und
- $\delta: Q \times \Sigma \rightarrow Q$  eine deterministische Übergangsfunktion ist.

Funktionsweise:

- Der Automat startet im Zustand  $q_0$  und liest nacheinander Zeichen aus  $\Sigma$ .
- $\delta$  ordnet jedem Paar  $(q, c)$  **einen** neuen Zustand zu.
- Ist der neue Zustand in  $F$ , gibt der Automat das Signal „akzeptiert“.

Bei Transformation von NFA zu DFA mit Teilmengenkonstruktion können aus  $k$  Zuständen des NFA bis zu  $2^k$  Zustände des DFA entstehen. Erfreulicherweise ändert sich die Anzahl der Zustände zwischen NFA und DFA bei der exakten Mustersuche **nicht**.

Funktionsweise:

- Der Automat startet im Zustand  $q_0$  und liest nacheinander Zeichen aus  $\Sigma$ .
- $\delta$  ordnet jedem Paar  $(q, c)$  **einen** neuen Zustand zu.
- Ist der neue Zustand in  $F$ , gibt der Automat das Signal „akzeptiert“.

Bei Transformation von NFA zu DFA mit Teilmengenkonstruktion können aus  $k$  Zuständen des NFA bis zu  $2^k$  Zustände des DFA entstehen. Erfreulicherweise ändert sich die Anzahl der Zustände zwischen NFA und DFA bei der exakten Mustersuche **nicht**.

Notation:  $P[i..j] := P[i : j + 1] = (P[i], \dots, P[j])$

## Theorem

*Sei  $A$  die aktive Zustandsmenge des Pattern-Matching-NFA.  
Sei  $a^* := \max A$ . Dann ist  $A$  durch  $a^*$  eindeutig bestimmt.  
Der äquivalente DFA hat genauso viele Zustände wie der NFA.*

Beispiel **abbab**:

$$a^* = -1: A = \{-1\}$$

$$a^* = 0 : A = \{-1, 0\}$$

$$a^* = 1 : A = \{-1, 1\}$$

$$a^* = 2 : A = \{-1, 2\}$$

$$a^* = 3 : A = \{-1, 0, 3\}$$

$$a^* = 4 : A = \{-1, 1, 4\}$$

# Beweis ( $A$ ist durch $a^*$ bestimmt)

- $a^*$  bestimmt die letzten  $a^* + 1$  gelesenen Textzeichen als  $P[0..a^*]$ .

# Beweis ( $A$ ist durch $a^*$ bestimmt)

- $a^*$  bestimmt die letzten  $a^* + 1$  gelesenen Textzeichen als  $P[0..a^*]$ .
- Zustand  $q < a^*$  ist genau dann auch aktiv, wenn die letzten  $q + 1$  gelesenen Zeichen gleich  $P[0..q]$  und gleich dem  $(q + 1)$ -Suffix von  $P[..a^*]$  sind, wenn also  $P[a^* - q..a^*] = P[..q]$ .

# Beweis ( $A$ ist durch $a^*$ bestimmt)

- $a^*$  bestimmt die letzten  $a^* + 1$  gelesenen Textzeichen als  $P[0..a^*]$ .
- Zustand  $q < a^*$  ist genau dann auch aktiv, wenn die letzten  $q + 1$  gelesenen Zeichen gleich  $P[0..q]$  und gleich dem  $(q + 1)$ -Suffix von  $P[..a^*]$  sind, wenn also  $P[a^* - q..a^*] = P[..q]$ .
- Da es also zu jedem  $a^*$  nur eine mögliche Zustandsmenge  $A$  mit  $a^* = \max A$  gibt, hat der DFA auf jeden Fall nicht mehr Zustände als der NFA.

# Beweis ( $A$ ist durch $a^*$ bestimmt)

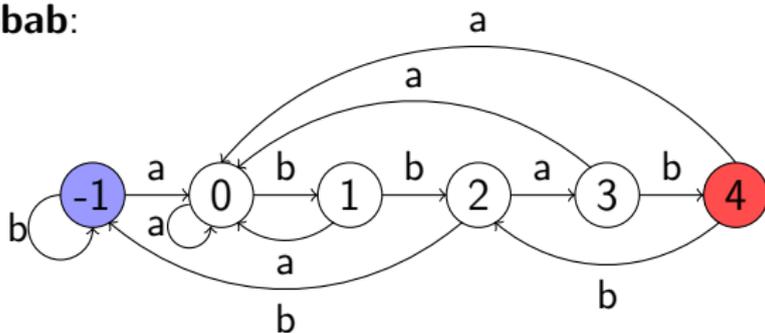
- $a^*$  bestimmt die letzten  $a^* + 1$  gelesenen Textzeichen als  $P[0..a^*]$ .
- Zustand  $q < a^*$  ist genau dann auch aktiv, wenn die letzten  $q + 1$  gelesenen Zeichen gleich  $P[0..q]$  und gleich dem  $(q + 1)$ -Suffix von  $P[..a^*]$  sind, wenn also  $P[a^* - q..a^*] = P[..q]$ .
- Da es also zu jedem  $a^*$  nur eine mögliche Zustandsmenge  $A$  mit  $a^* = \max A$  gibt, hat der DFA auf jeden Fall nicht mehr Zustände als der NFA.
- Da aber auch jeder NFA-Zustand vom Startzustand aus erreichbar ist, hat der DFA auch nicht weniger Zustände als der NFA. □

Formal ergibt sich der DFA wie folgt:

- $Q = \{-1, 0, \dots, m - 1\}$  ( $m + 1$  Zustände)
- $q_0 = -1$
- $\Sigma$  ist das Alphabet des Textes und Patterns
- $F = \{m - 1\}$
- Übergangsfunktion  $\delta : Q \times \Sigma \rightarrow Q$ :  
Berechne eindeutige NFA-Zustandsmenge  $A(q)$  mit  $q = \max A(q)$ . Wende hierauf  $\Delta$  für jedes  $c$  an; extrahiere Maximum:

$$\delta(q, c) := \max \Delta(A(q), c)$$

DFA für **abbab**:

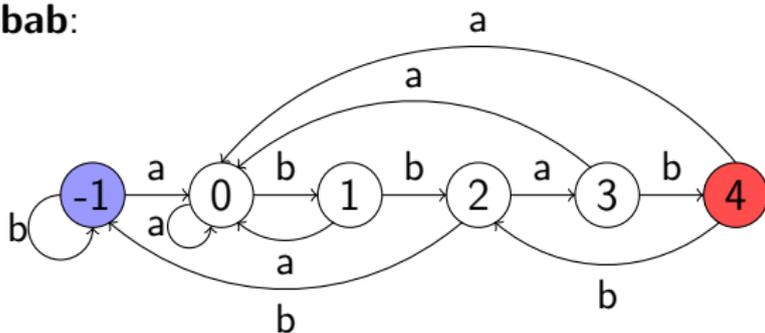


Beispiel:  $q = 3$ ,  $a^* = 3$ ,  $A = \{-1, 0, 3\}$ ,  $c = a$ :

$$\left. \begin{array}{l}
 \Delta(-1, a) = \{-1, 0\} \\
 \Delta(0, a) = \{\} \\
 \Delta(3, a) = \{\}
 \end{array} \right\} \text{ Vereinigung: } \{-1, 0\}$$

Also  $\delta(3, a) = 0$ .

DFA für **abbab**:



Beispiel:  $q = 3$ ,  $a^* = 3$ ,  $A = \{-1, 0, 3\}$ ,  $c = b$ :

$$\left. \begin{array}{l} \Delta(-1, b) = \{-1\} \\ \Delta(0, b) = \{1\} \\ \Delta(3, b) = \{4\} \end{array} \right\} \text{Vereinigung: } \{-1, 1, 4\}$$

Also  $\delta(3, b) = 4$ .

```
1 def DFA_matcher(P, T):
2     delta = DFA_delta_table(P)
3     (q, m, n) = (-1, len(P), len(T))
4     for i in range(n):
5         q = delta(q, T[i])
6         if q == m - 1:
7             yield (i-m+1, i+1)
8
9 def DFA(P, T):
10     for s, e in DFA_matcher(P, T):
11         print("found:", s, e)
```

Exakte Mustersuche mit einem DFA ist relativ effizient, sobald der Automat ( $\delta$ -Funktion) erstellt wurde.

Knuth, Morris und Pratt (“KMP”) beschreiben einen Algorithmus, der die Übergangsfunktion kompakt darstellt.

Merkmal	<i>DFA</i>	<i>KMP</i>
Zeit zur Konstruktion der $\delta$ -Tabelle	$\mathcal{O}(m^2 \Sigma )$	$\mathcal{O}(m)$
Speicherplatz der $\delta$ -Tabelle	$\mathcal{O}(m \Sigma )$	$\mathcal{O}(m)$
Zeit für Mustersuche im Text	$\mathcal{O}(n)$	$\mathcal{O}(n)$

**Idee:** Wir simulieren den DFA-Übergang, indem wir den NFA-Übergang simulieren und  $A(q)$  on-the-fly berechnen.

$A(q)$  ist die eindeutige Zustandsmenge, die  $q$  als maximales Element hat. Welches ist das nächstkleinere Element von  $A(q)$ ?

**Idee:** Wir simulieren den DFA-Übergang, indem wir den NFA-Übergang simulieren und  $A(q)$  on-the-fly berechnen.

$A(q)$  ist die eindeutige Zustandsmenge, die  $q$  als maximales Element hat. Welches ist das nächstkleinere Element von  $A(q)$ ?

## Definition (Ips-Funktion)

Zu  $P \in \Sigma^m$  definieren wir  $\text{Ips} : \{0, \dots, m-1\} \rightarrow \mathbb{N}$ :

$$\text{Ips}(q) := \max\{|s| \leq q : s \text{ ist Präfix von } P \text{ und Suffix von } P[..q]\}.$$

(“längstes **P**räfix von  $P$ , das echtes **S**uffix von  $P[..q]$  ist”)

- $A(q)$ : aktive NFA-Zustände, wenn  $q = \max A(q)$
- $\text{lps}(q) = \max\{|s| \leq q : s \text{ Präfix von } P \text{ und Suffix von } P[..q]\}$

- $A(q)$ : aktive NFA-Zustände, wenn  $q = \max A(q)$
- $\text{lps}(q) = \max\{|s| \leq q : s \text{ Präfix von } P \text{ und Suffix von } P[..q]\}$
- Zustand  $q$  aktiv  $\Leftrightarrow$  letzte  $q + 1$  Textzeichen sind  $P[0..q]$
- $p$  und  $q$  aktiv,  $p < q$ : Es muss  $P[q - p..q] = P[..p]$  sein

## Lemma

$$A(q) = \{ q, \text{lps}(q) - 1, \text{lps}(\text{lps}(q) - 1) - 1, \dots, -1 \}$$

# Die lps-Funktion

- $A(q)$ : aktive NFA-Zustände, wenn  $q = \max A(q)$
- $\text{lps}(q) = \max\{|s| \leq q : s \text{ Präfix von } P \text{ und Suffix von } P[..q]\}$
- Zustand  $q$  aktiv  $\Leftrightarrow$  letzte  $q + 1$  Textzeichen sind  $P[0..q]$
- $p$  und  $q$  aktiv,  $p < q$ : Es muss  $P[q - p..q] = P[..p]$  sein

## Lemma

$$A(q) = \{ q, \text{lps}(q) - 1, \text{lps}(\text{lps}(q) - 1) - 1, \dots, -1 \}$$

$q$	0	1	2	3	4	5	6
$P[q]$	a	b	a	b	a	c	a
$\text{lps}[q]$	0	0	1	2	3	0	1

# Simulation der $\delta$ -Funktion mit lps

Beginnend in  $q$ , wenn das gelesene Zeichen  $c$  nicht das nächste Zeichen in  $P$  ist, gehe zurück auf den nächst kleineren Zustand in  $A(q)$ , bis entweder  $c$  passt oder  $A(q)$  erschöpft ist.

```
1 def DFA_delta_lps(q, c, P, lps):
2     m = len(P)
3     while q == m-1 or (P[q+1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q+1] == c:
6         q += 1
7     return q
```

# Laufzeitanalyse der Mustersuche mit KMP



## Lemma

*Die Laufzeit des Knuth-Morris-Pratt-Algorithmus auf einem Text der Länge  $n$  ist  $\mathcal{O}(n)$ , wenn die  $lps$ -Funktion des Musters vorliegt.*

- Aufruf von `DFA_delta_lps` kann  $\mathcal{O}(m)$  Zeit kosten (while).
- `DFA_delta_lps` wird  $\mathcal{O}(n)$  mal aufgerufen.
- **Gesamtzahl** der while-Durchläufe ist beschränkt:  
 $q$  wird kleiner!
- Pro Funktionsaufruf wird  $q$  höchstens um 1 erhöht.
- $q$  immer zwischen  $-1$  und  $m$

## Amortisierte Analyse!

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$T =$ 

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
---	---	---	---	---	---	---

$q = -1$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$T =$ 

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
---	---	---	---	---	---	---

$q = -1$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$T =$ 

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
---	---	---	---	---	---	---

$q = -1$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$T =$ 

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
---	---	---	---	---	---	---

$q = -1$

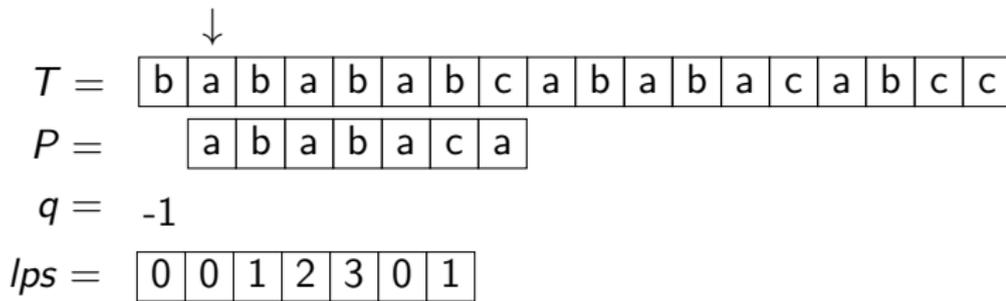
$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

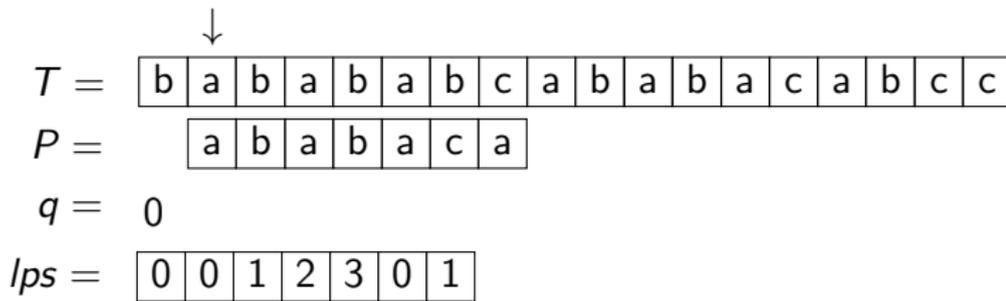
↓

$T =$	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						
$q =$	-1																		
$lps =$	0	0	1	2	3	0	1												

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



$T =$ 

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
---	---	---	---	---	---	---

$q = 0$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



$T =$ 

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
---	---	---	---	---	---	---

$q = 0$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



$T =$ 

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
---	---	---	---	---	---	---

$q = 1$

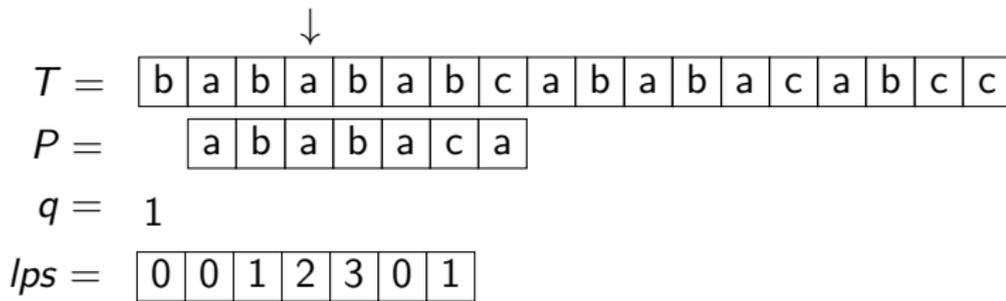
$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

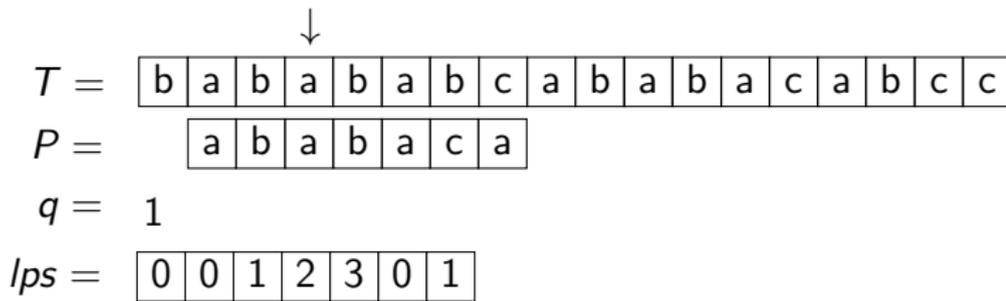
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

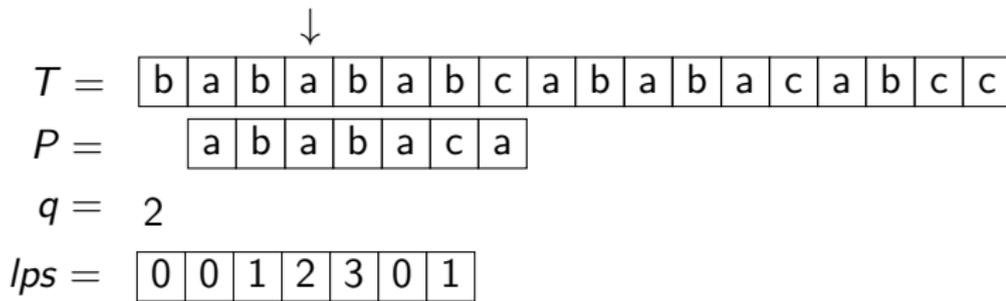
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



$T =$ 

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
---	---	---	---	---	---	---

$q = 2$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



$T =$ 

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
---	---	---	---	---	---	---

$q = 2$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



$T =$ 

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
---	---	---	---	---	---	---

$q = 3$

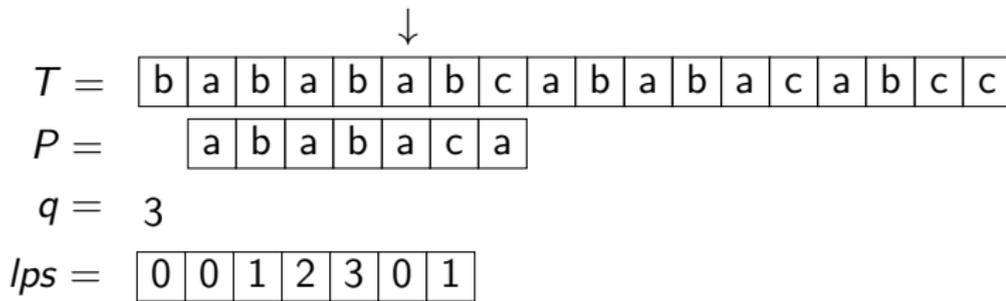
$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

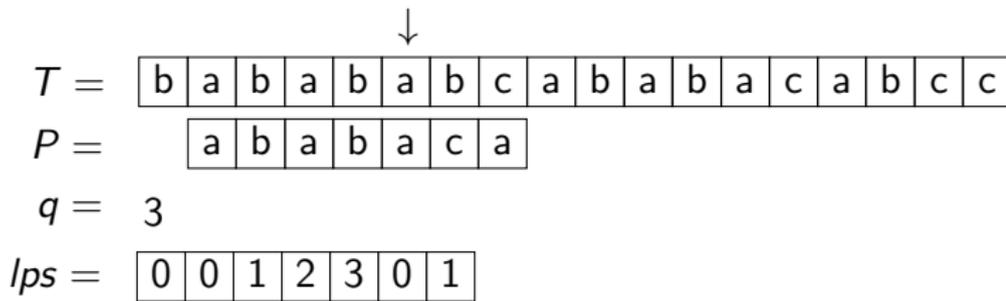
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$T =$ 

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
---	---	---	---	---	---	---

$q = 4$

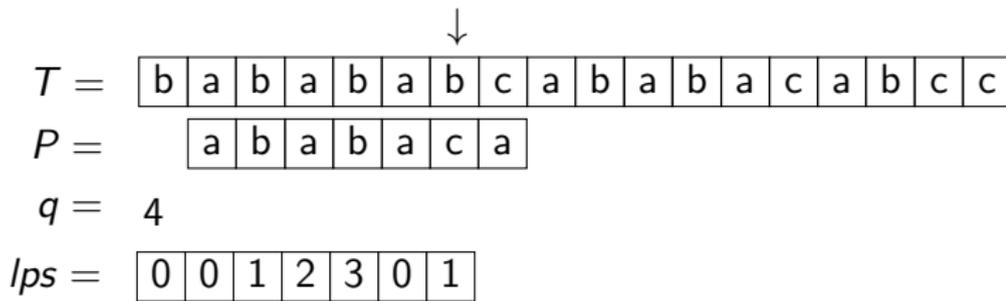
$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

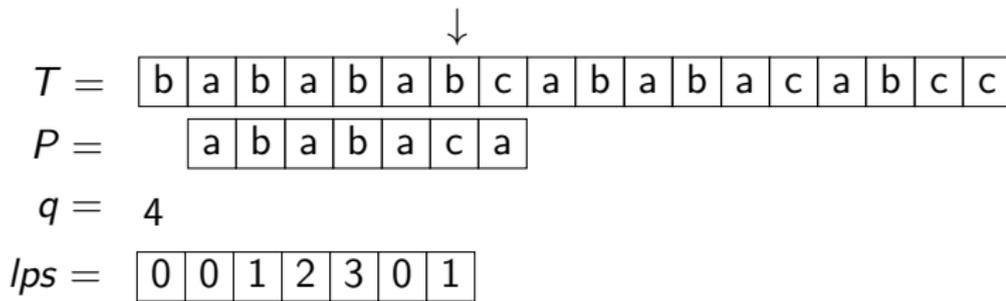
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

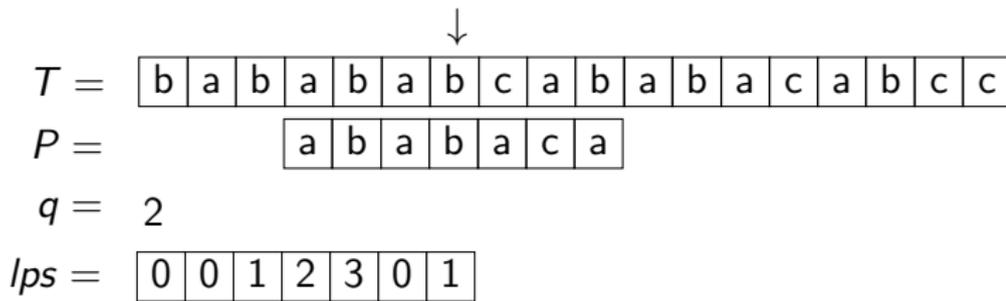
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

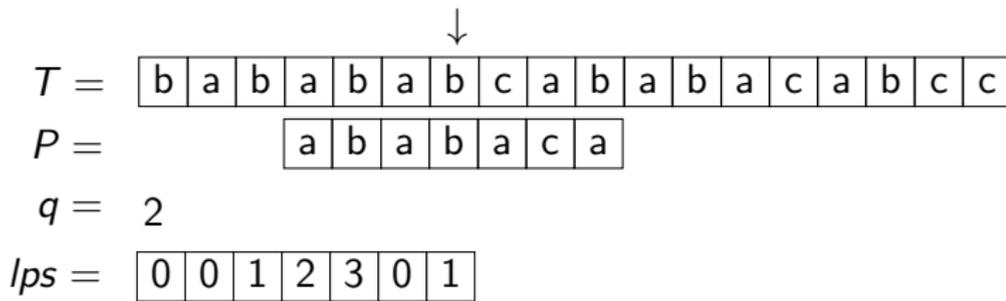
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

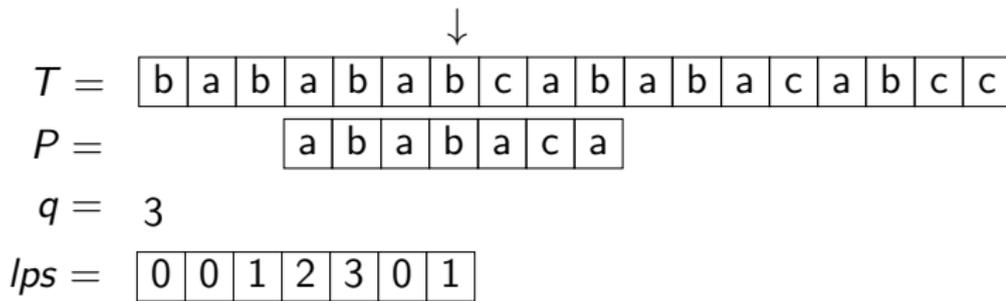
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

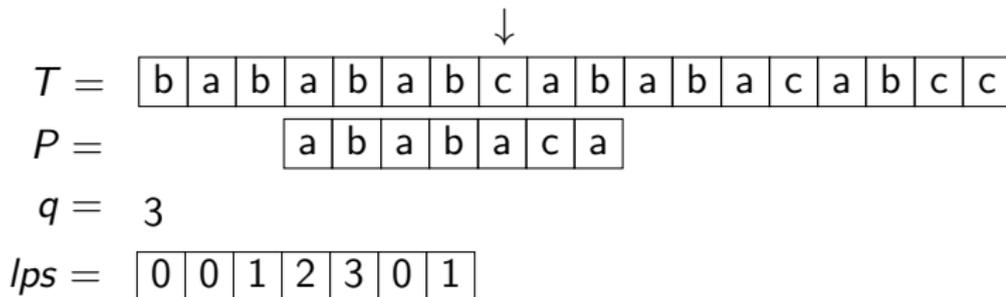
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

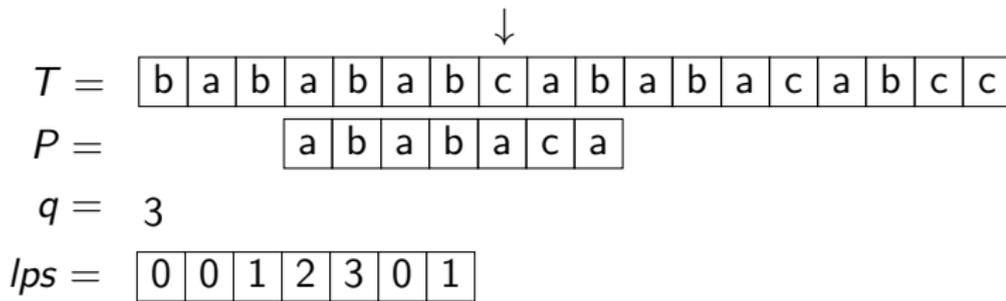
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

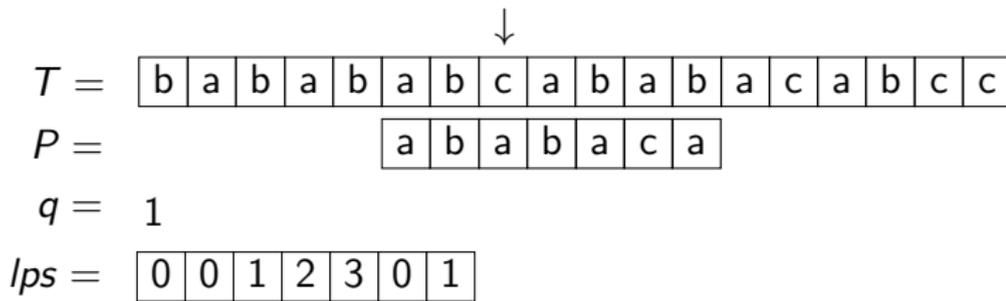
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

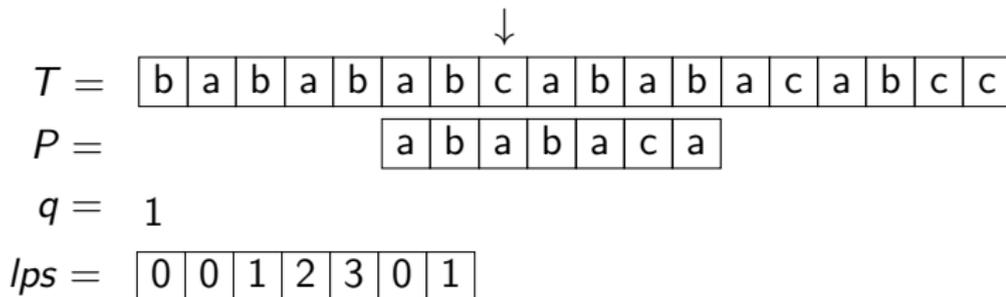
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

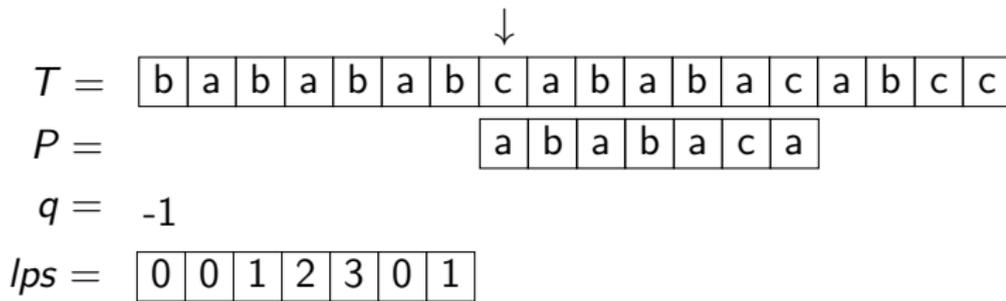
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

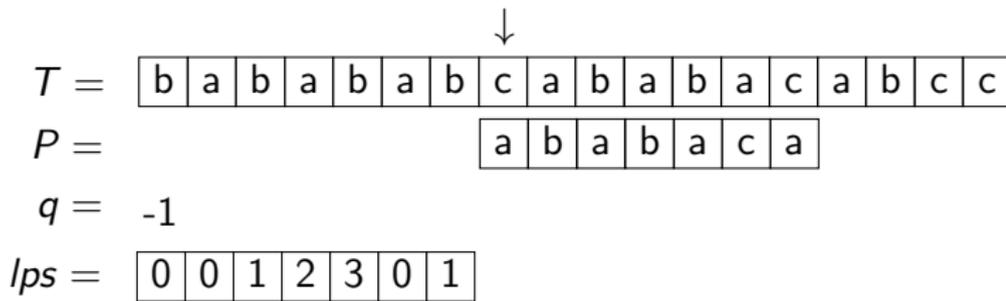
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

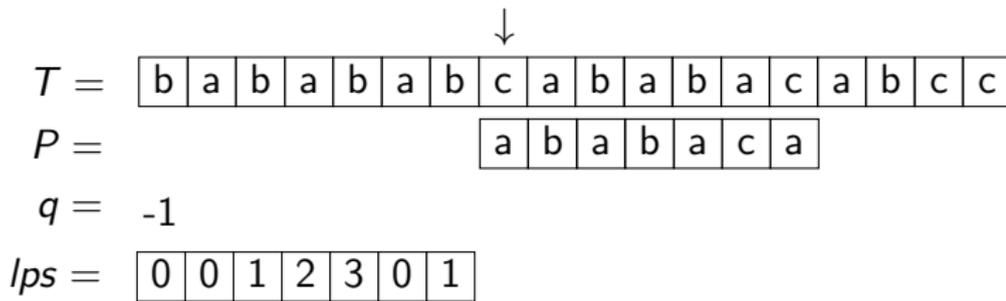
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

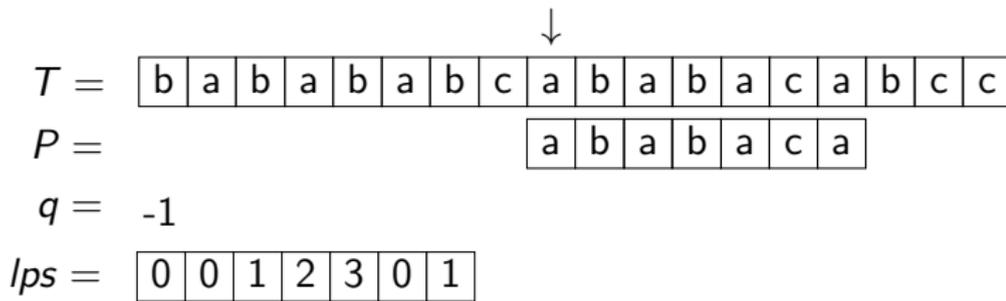
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

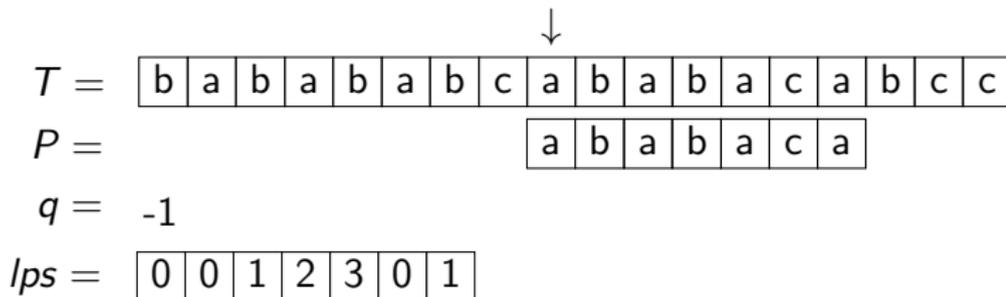
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

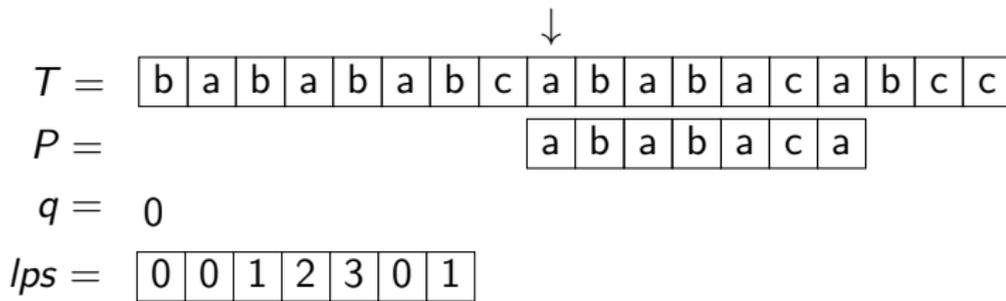
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

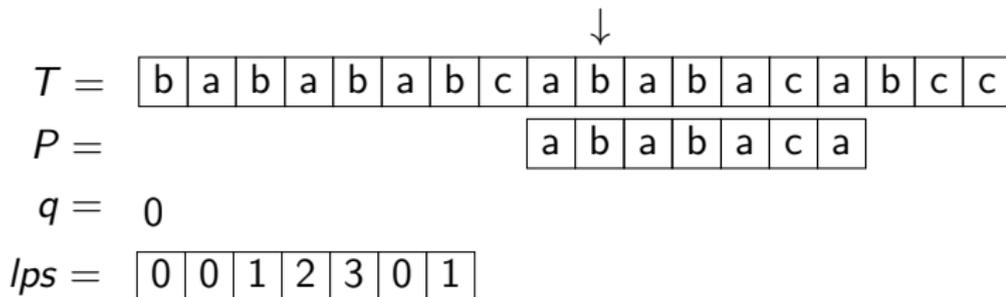
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

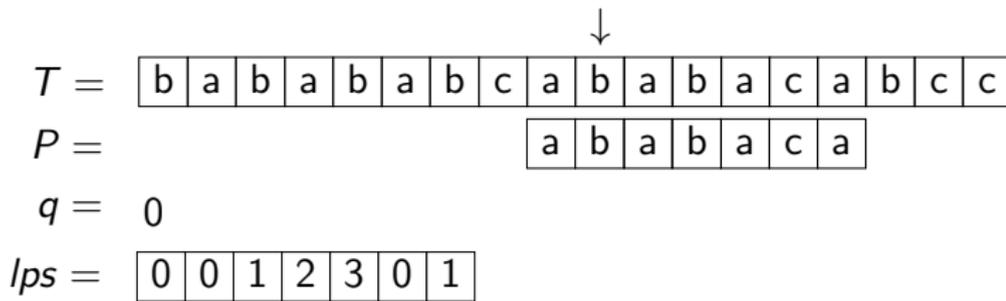
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

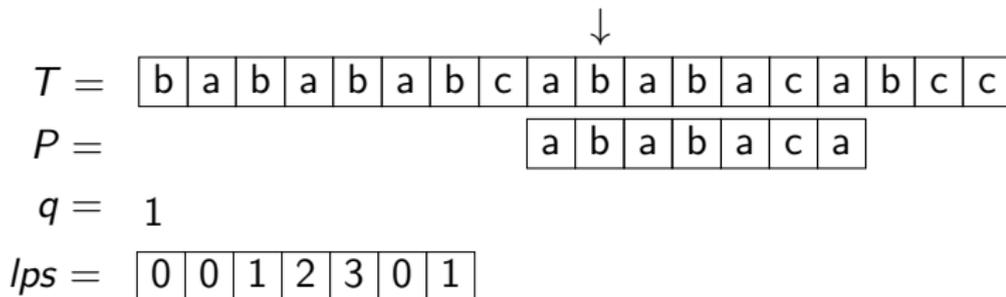
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

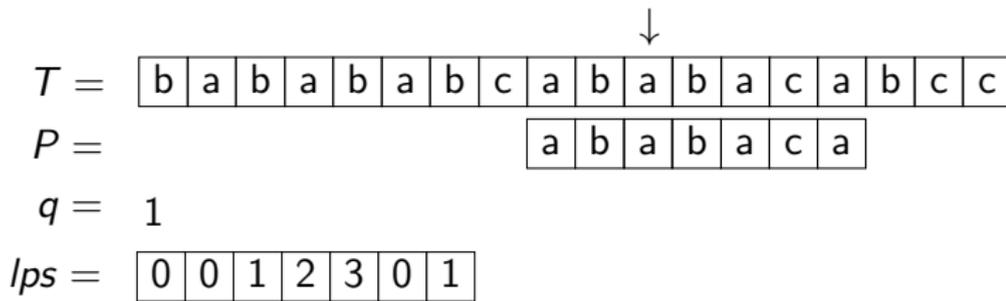
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

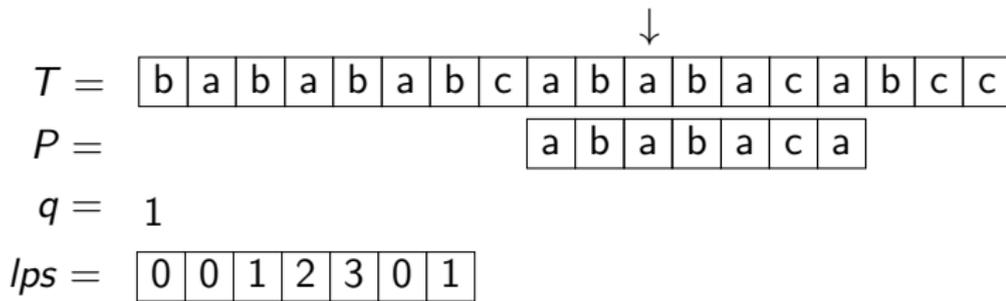
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

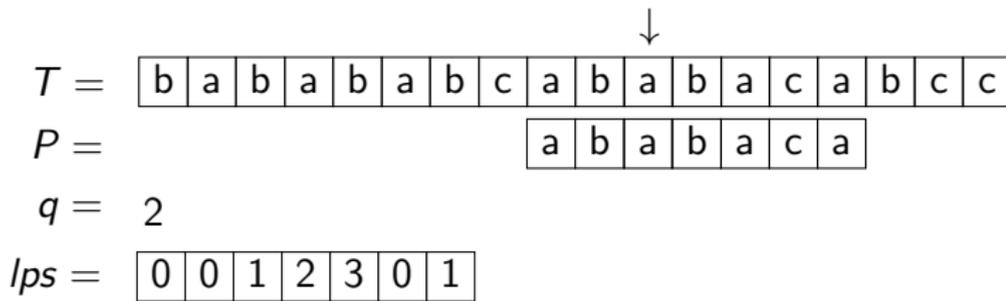
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

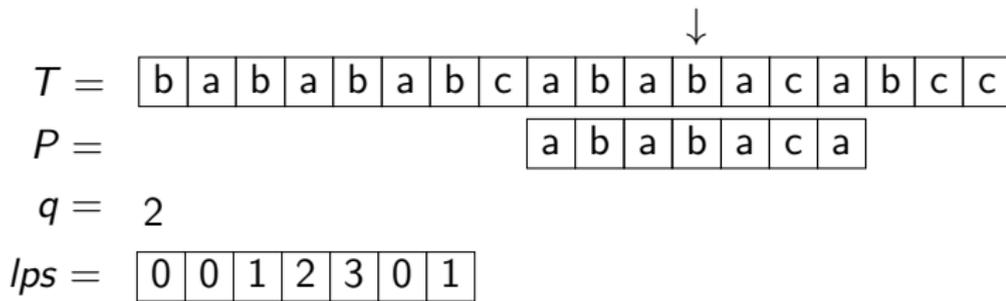
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

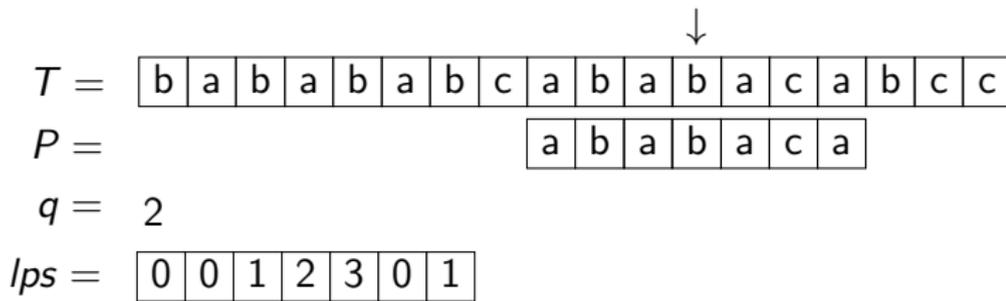
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

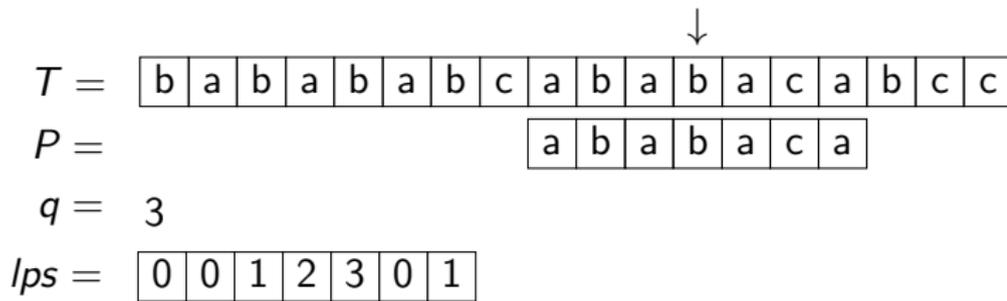
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

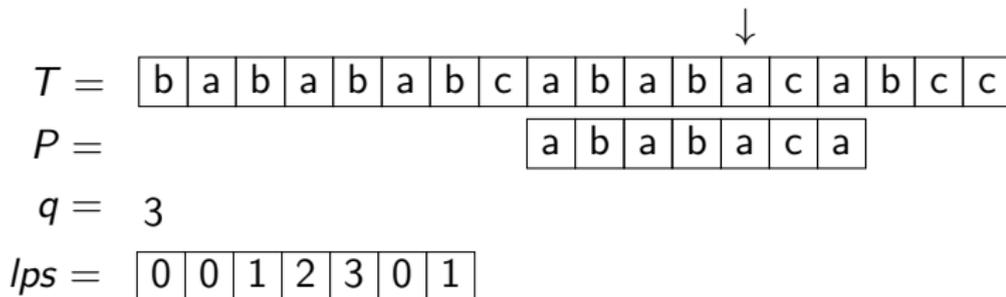
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

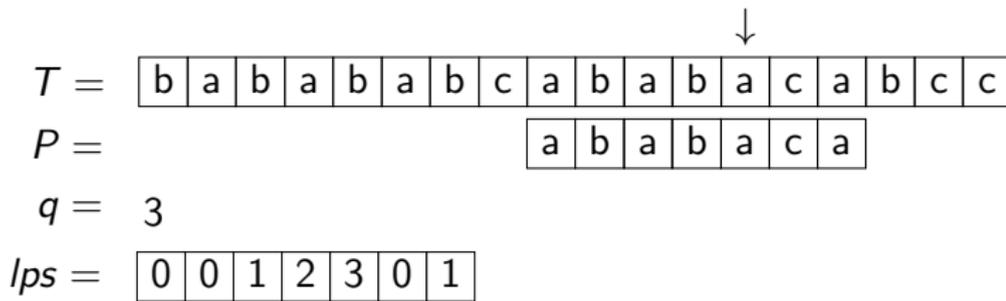
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

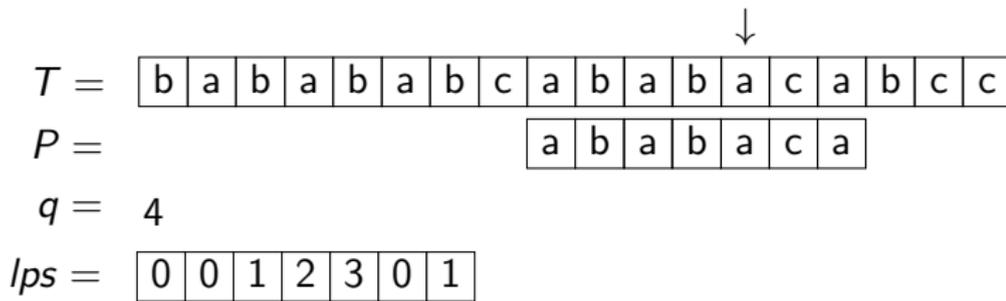
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

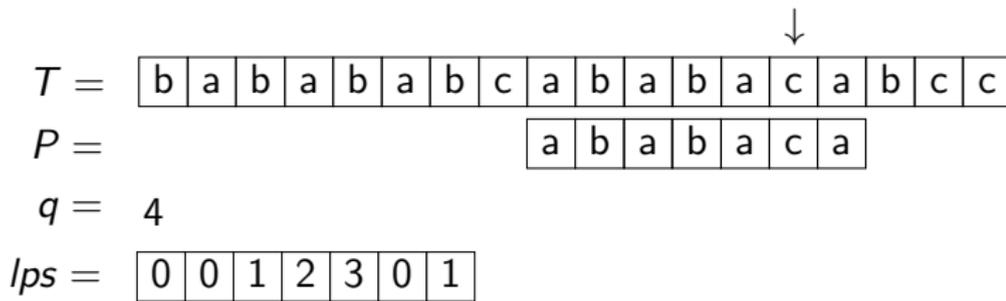
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

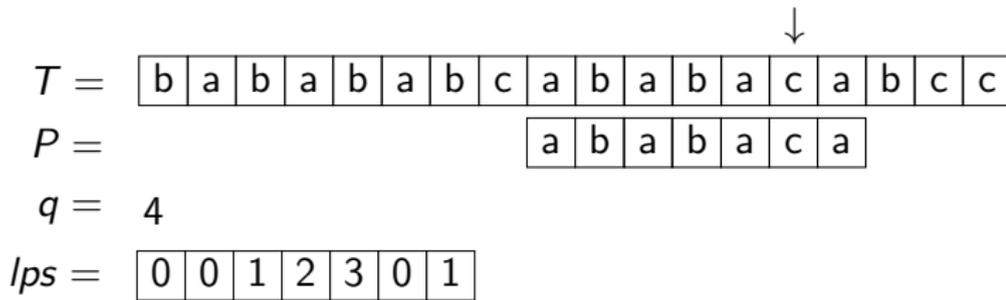
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

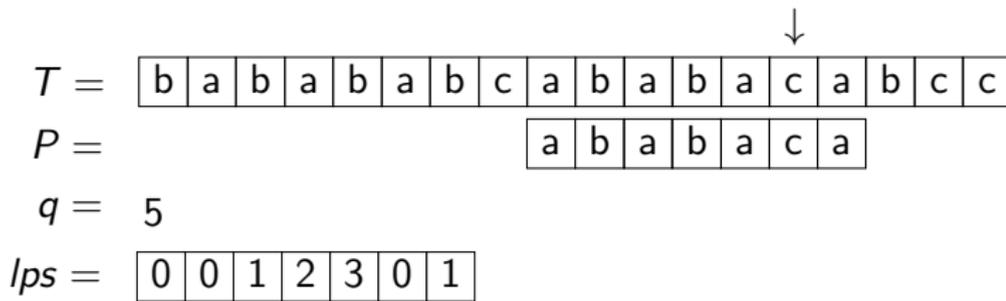
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

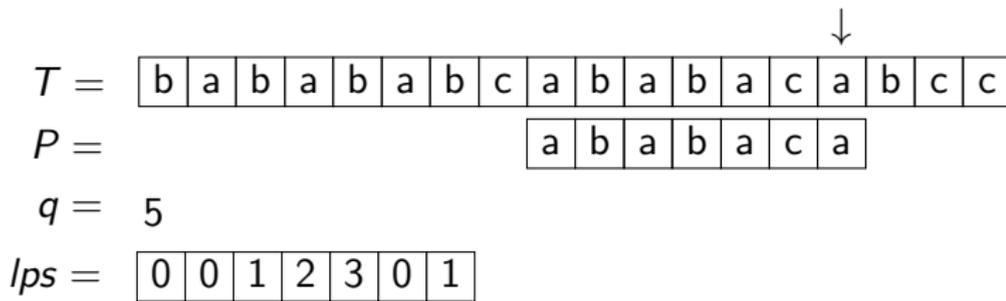
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

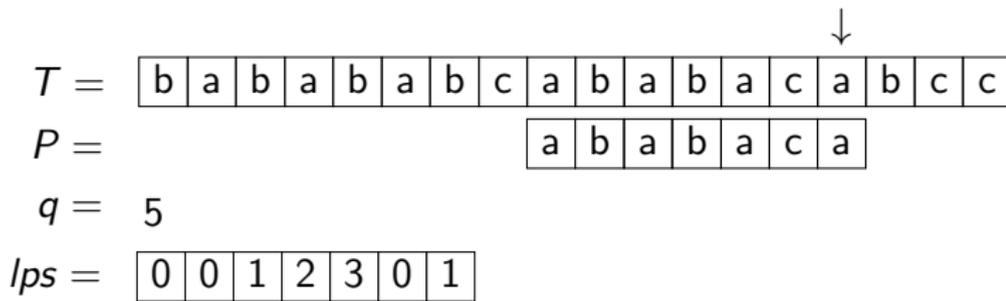
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$T =$ 

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
---	---	---	---	---	---	---

$q = 6$

$lps =$ 

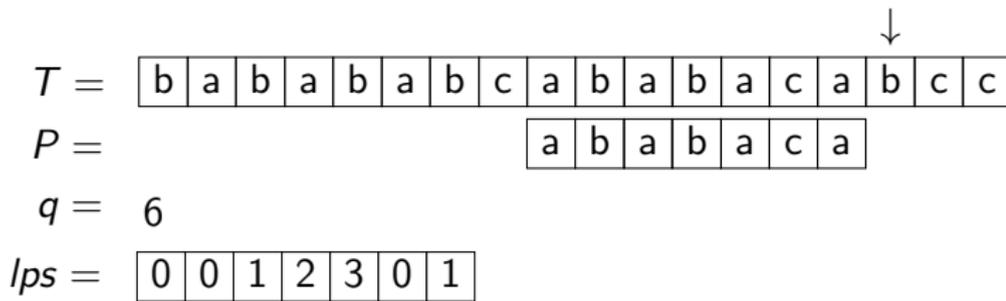
0	0	1	2	3	0	1
---	---	---	---	---	---	---

 yield 9, 16

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

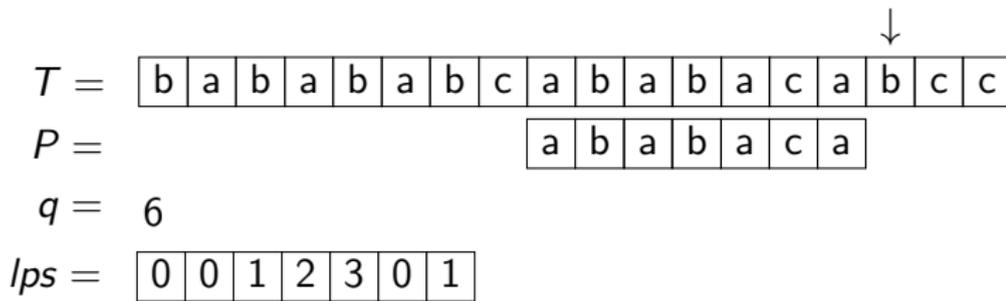
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

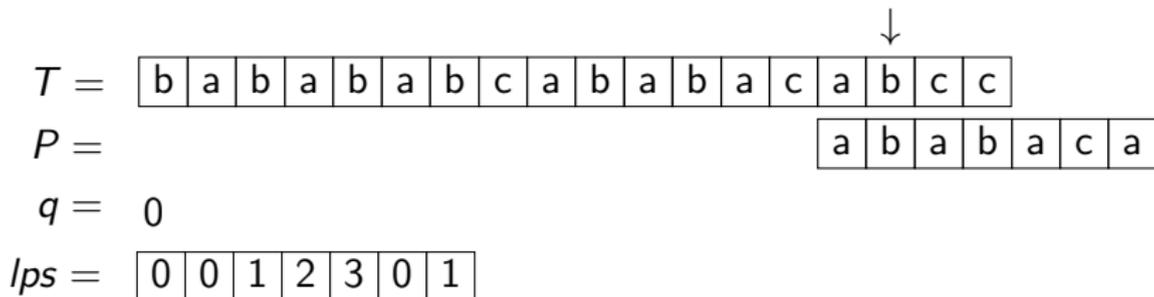
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

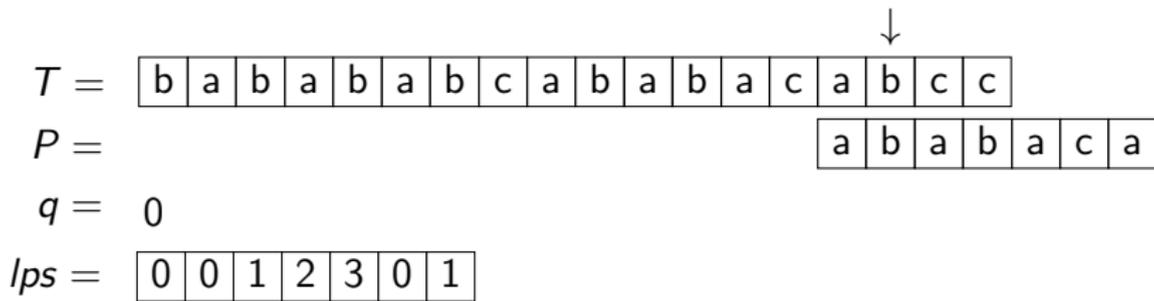
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

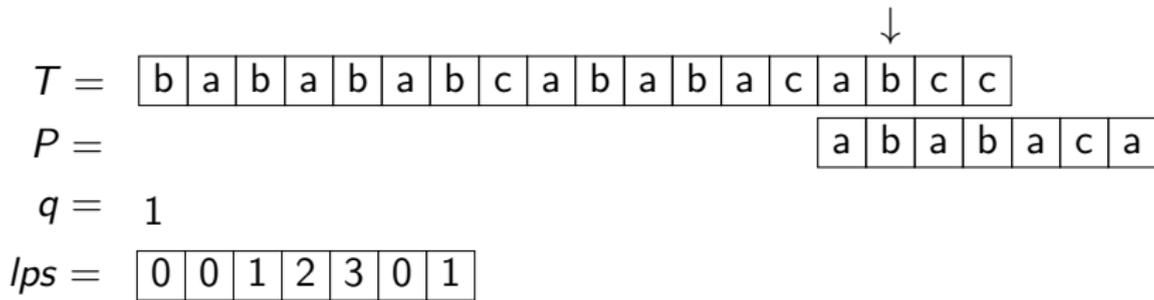
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

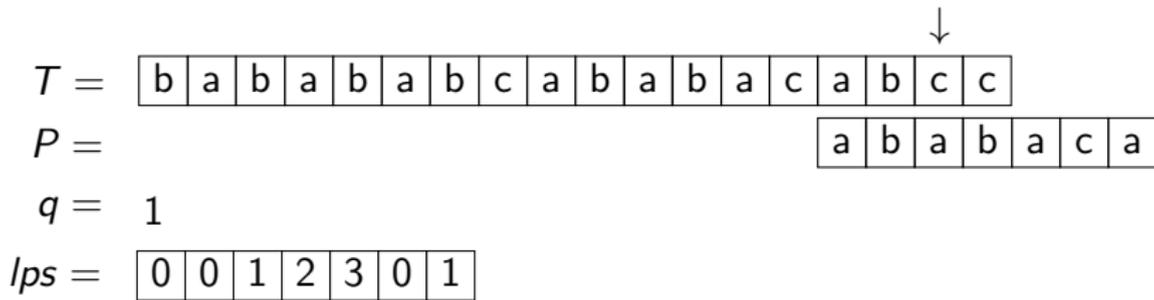
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

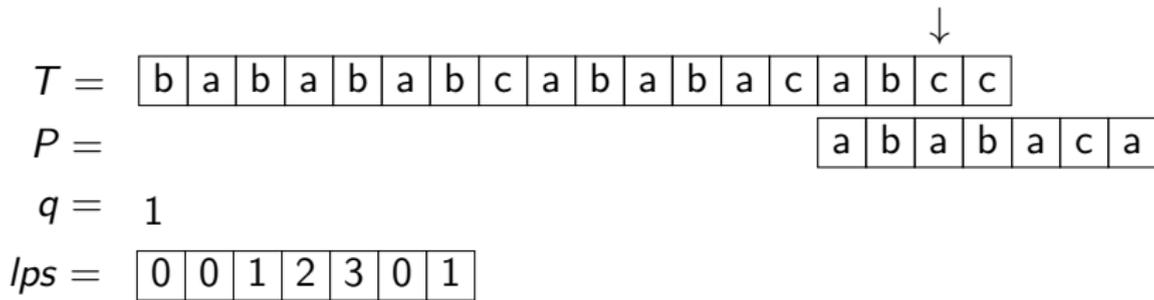
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

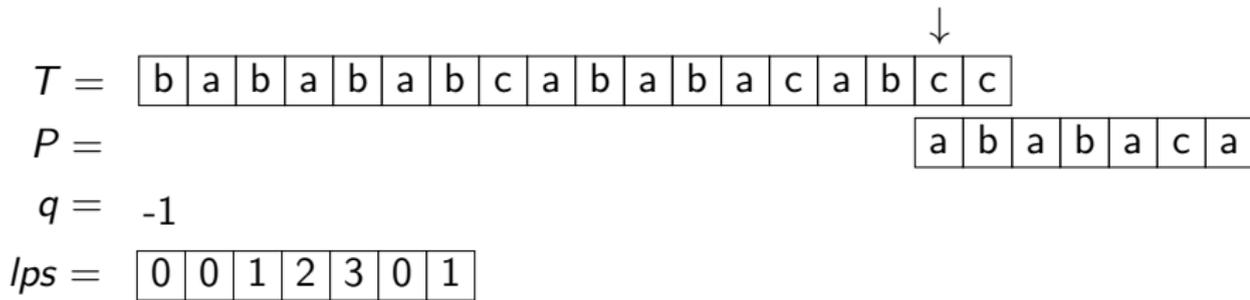
```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

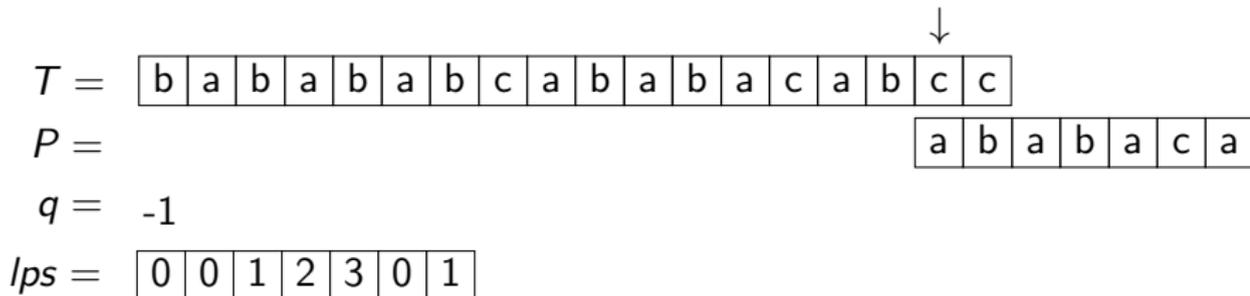
```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

$T =$ 

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
---	---	---	---	---	---	---

$q = -1$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

↓



# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

$T =$ 

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
---	---	---	---	---	---

$q = -1$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

↓

# Beispiel zur Mustersuche mit KMP

Mustersuche mit dem KMP:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

$T =$ 

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
---	---	---	---	---	---

$q = -1$

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

↓

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓  
P = a b a b a c a  
lps = 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

  
c = b  
q = -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

$c =$  b

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

$P =$  a b a b a c a  
          ↓  
 $lps =$ 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

  
 $c =$  b  
 $q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

$c =$  b

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	0	0	0	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	0	0	0	0
---	---	---	---	---	---	---

$c =$  b

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	0	0	0	0
---	---	---	---	---	---	---

$c =$  b

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$P =$     a b a b a c a

$lps =$ 

0	0	1	0	0	0	0
---	---	---	---	---	---	---

$c =$     b

$q =$     1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	0	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	0	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

$P =$  a b a b a c a  
                                  ↓  
 $lps =$ 

0	0	1	2	0	0	0
---	---	---	---	---	---	---

  
 $c =$  a  
 $q =$  2

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

$P =$  a b a b a c a

↓

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  2

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  2

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  c

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):
2     m = len(P)
3     while q == m - 1 or (P[q + 1] != c and q > -1):
4         q = lps[q] - 1
5     if P[q + 1] == c: q += 1
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  -1

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	0
---	---	---	---	---	---	---

$c =$  a

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

↓

$P =$  a b a b a c a

$lps =$ 

0	0	1	2	3	0	1
---	---	---	---	---	---	---

$c =$  a

$q =$  0

Erstellung der lps-Tabelle mit lps selbst:

```
1 def delta(q, c, P, lps):  
2     m = len(P)  
3     while q == m - 1 or (P[q + 1] != c and q > -1):  
4         q = lps[q] - 1  
5     if P[q + 1] == c: q += 1  
6     return q
```

Die (amortisierte) Laufzeit beträgt  $\mathcal{O}(m)$ .

- Bei einem DFA wird jedes Textzeichen nur einmal betrachtet.
- Zustandsaktualisierung ( $\delta$ -Funktion) entspricht Tabellen-Lookup: echtzeitfähig
- Der Knuth-Morris-Pratt-Algorithmus realisiert einen DFA für die exakte Mustersuche durch kompakte Repräsentierung der Übergangsfunktion  $\delta$  mit Hilfe der lps-Funktion.
- Laufzeiten für KMP durch amortisierte Analyse (nicht echtzeitfähig):
  - Konstruktion:  $\mathcal{O}(m)$
  - Suche:  $\mathcal{O}(n)$
- KMP ist cache-effizient, da der Text sequenziell gelesen wird.
- Dennoch ist KMP in der Praxis nie der schnellste Algorithmus!
- Interesse vor allem theoretisch und historisch.