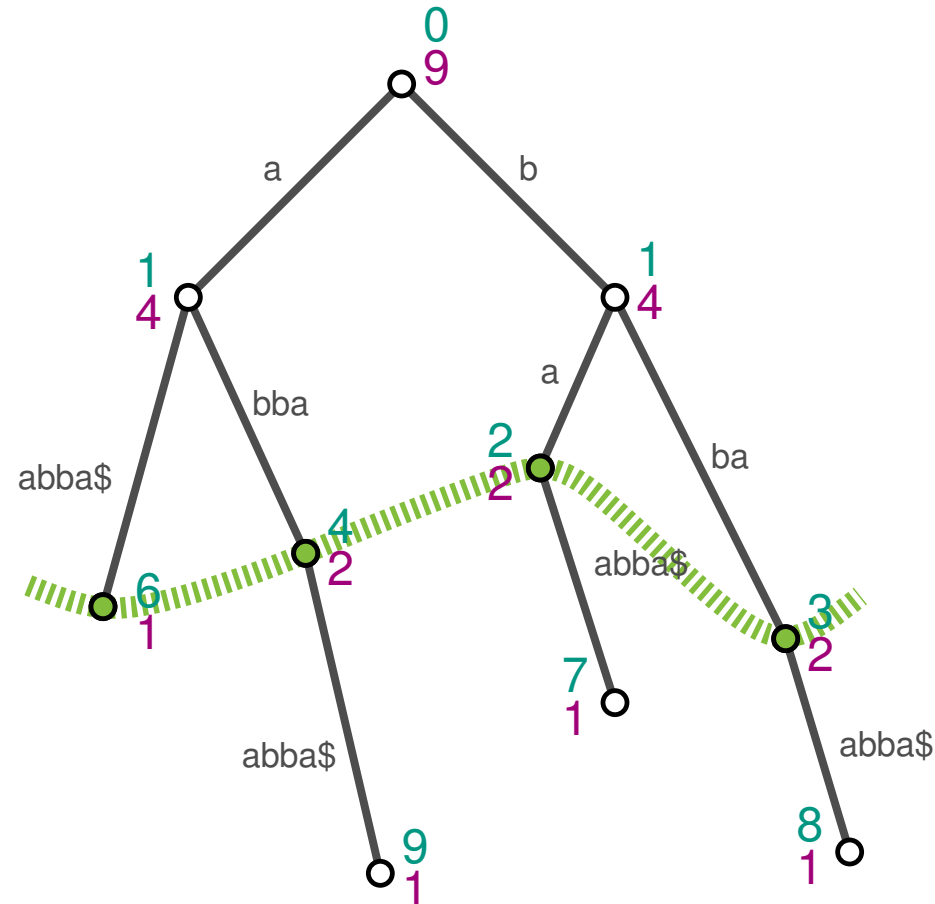


# Textindexierung Programmierprojekt

Abschlusspräsentation · 07.02.2022

Moritz Potthoff

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# Vorbereitung: Suffix Tree

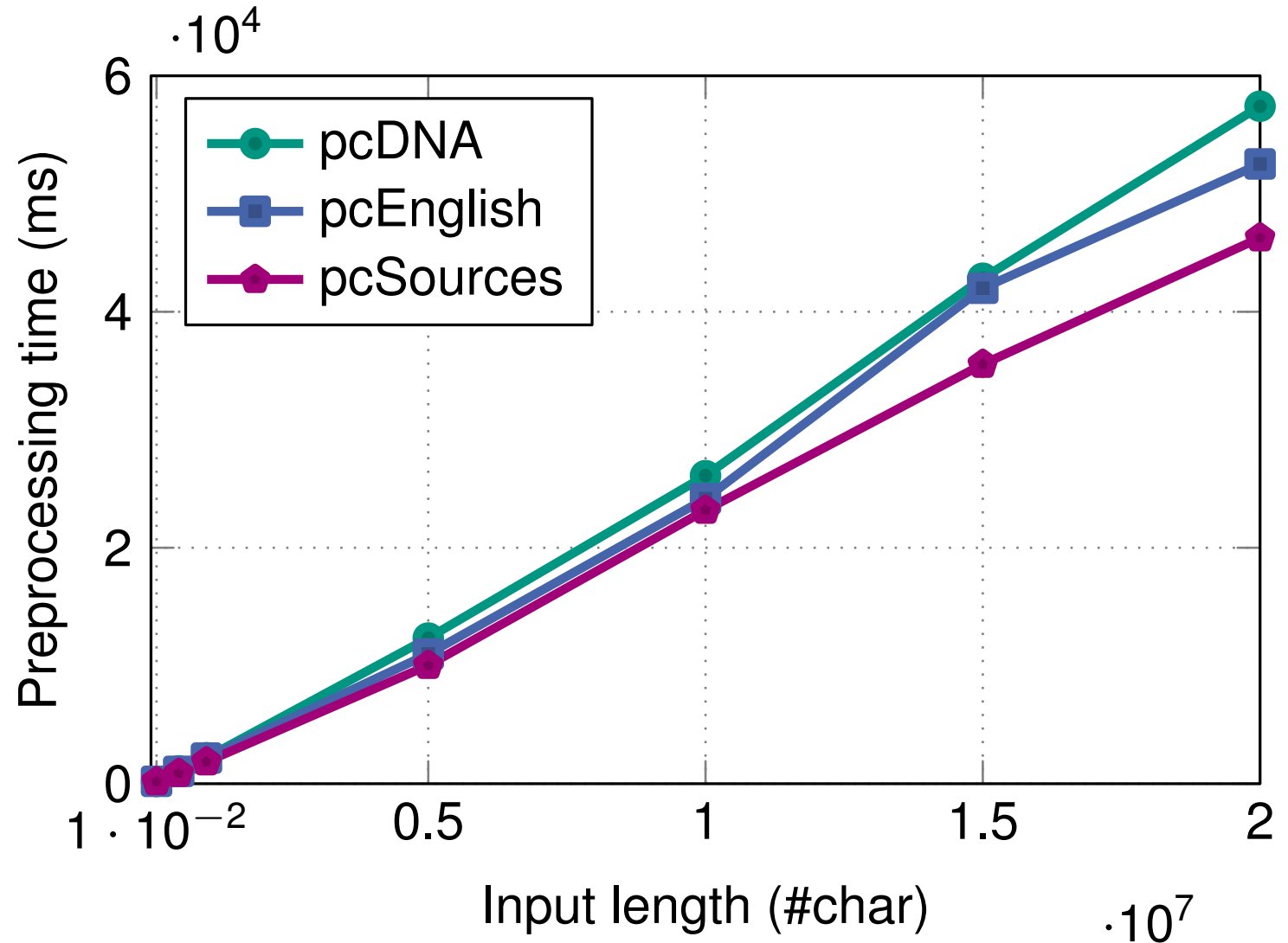
## Suffix Tree-Konstruktion:

- In  $\mathcal{O}(n)$  mittels  
**Ukkonens Algorithmus**
- Hoher Platzbedarf
- Lineare Laufzeit

# Vorbereitung: Suffix Tree

## Suffix Tree-Konstruktion:

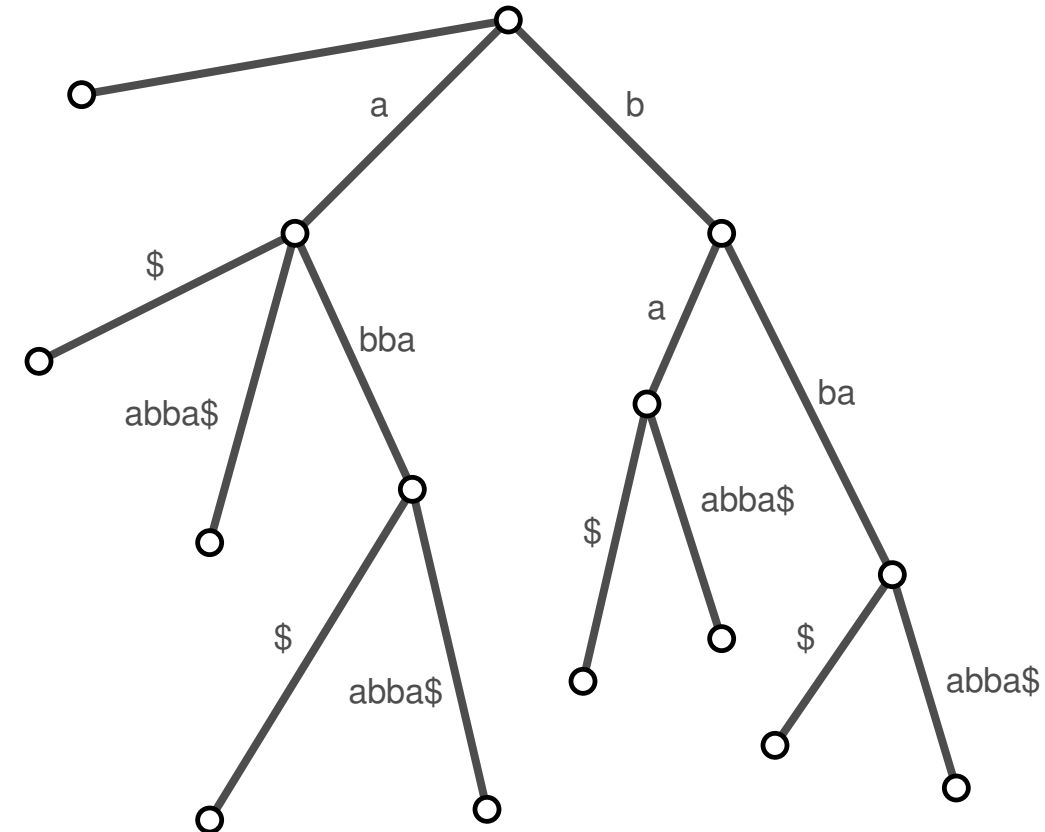
- In  $\mathcal{O}(n)$  mittels **Ukkonens Algorithmus**
- Hoher Platzbedarf
- Lineare Laufzeit



# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

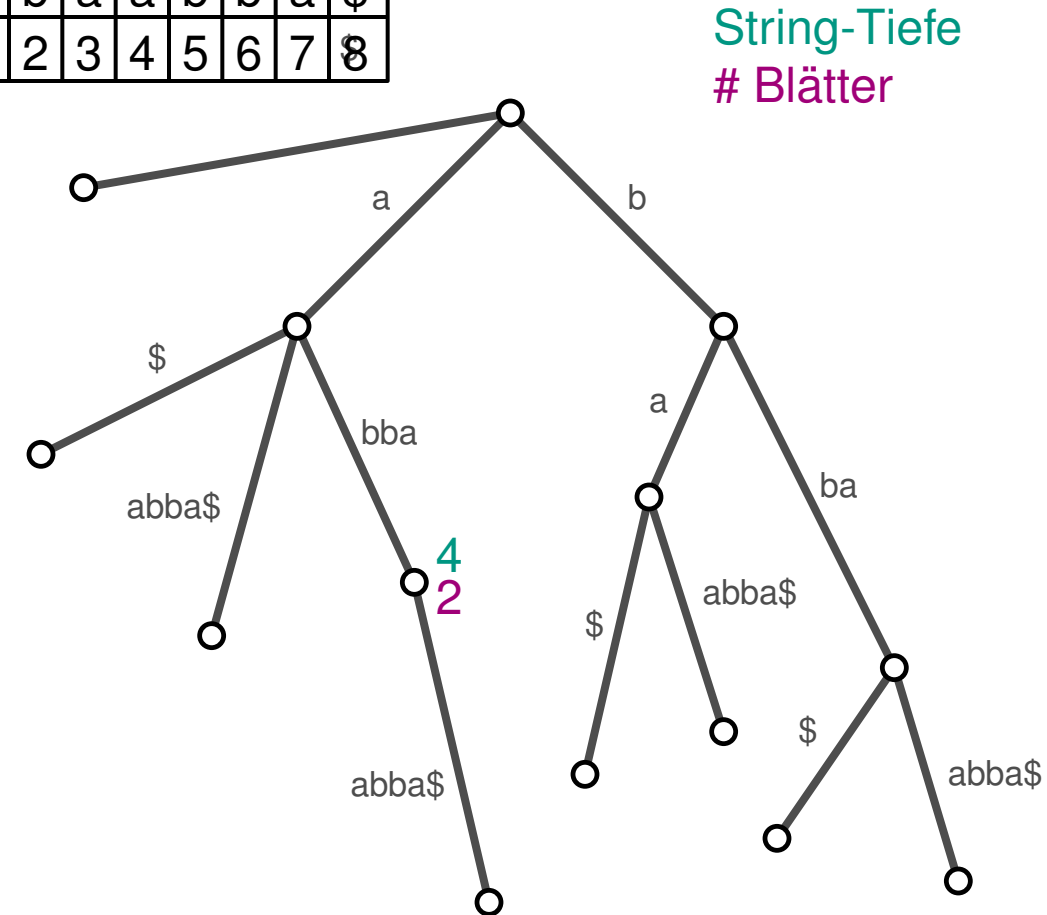
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

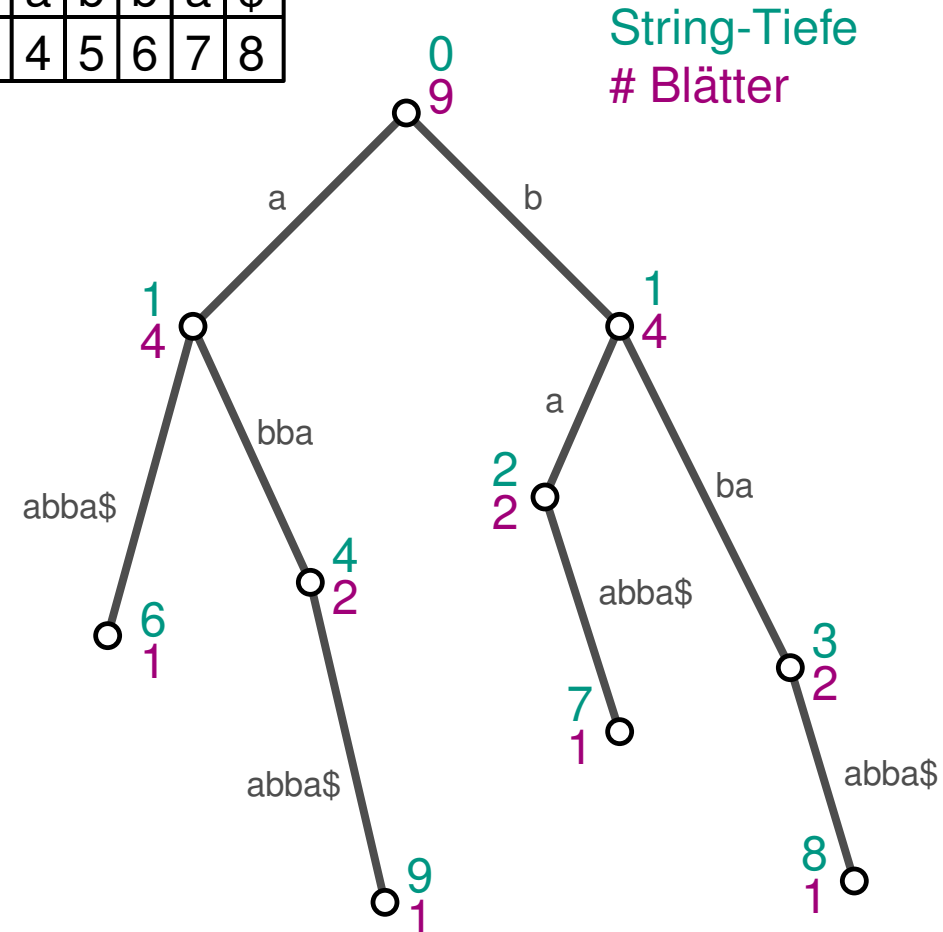
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

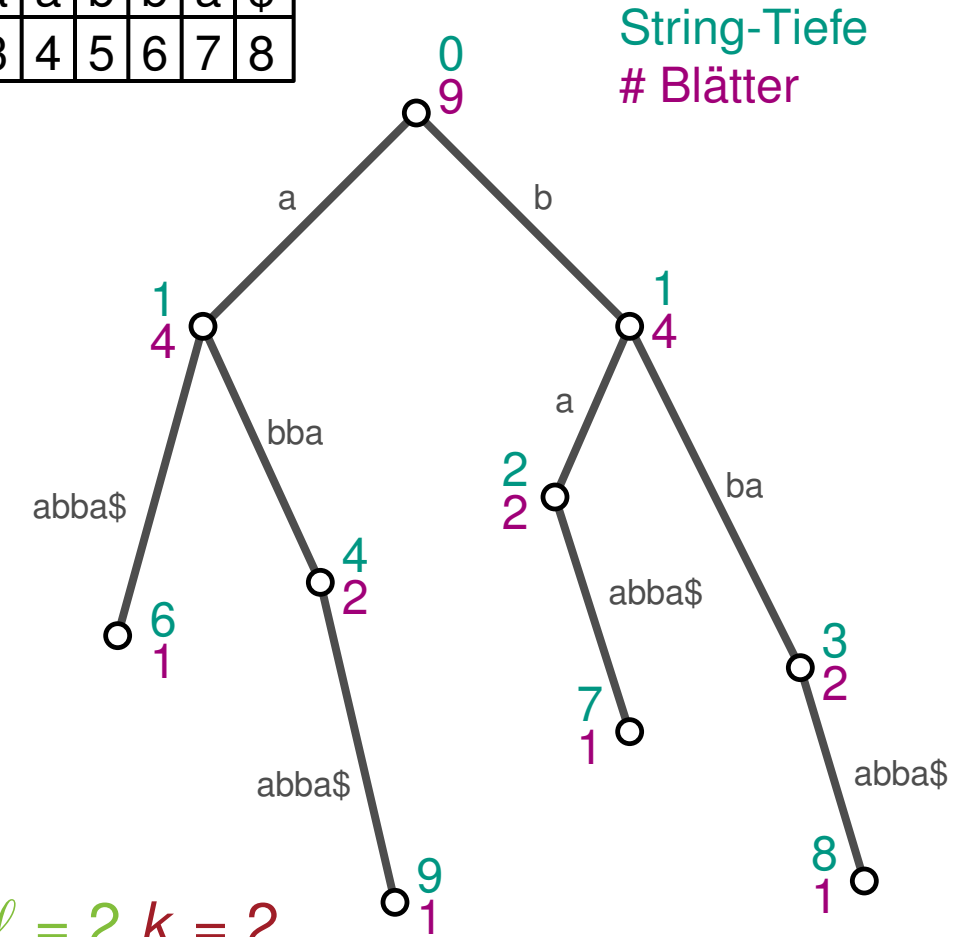
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

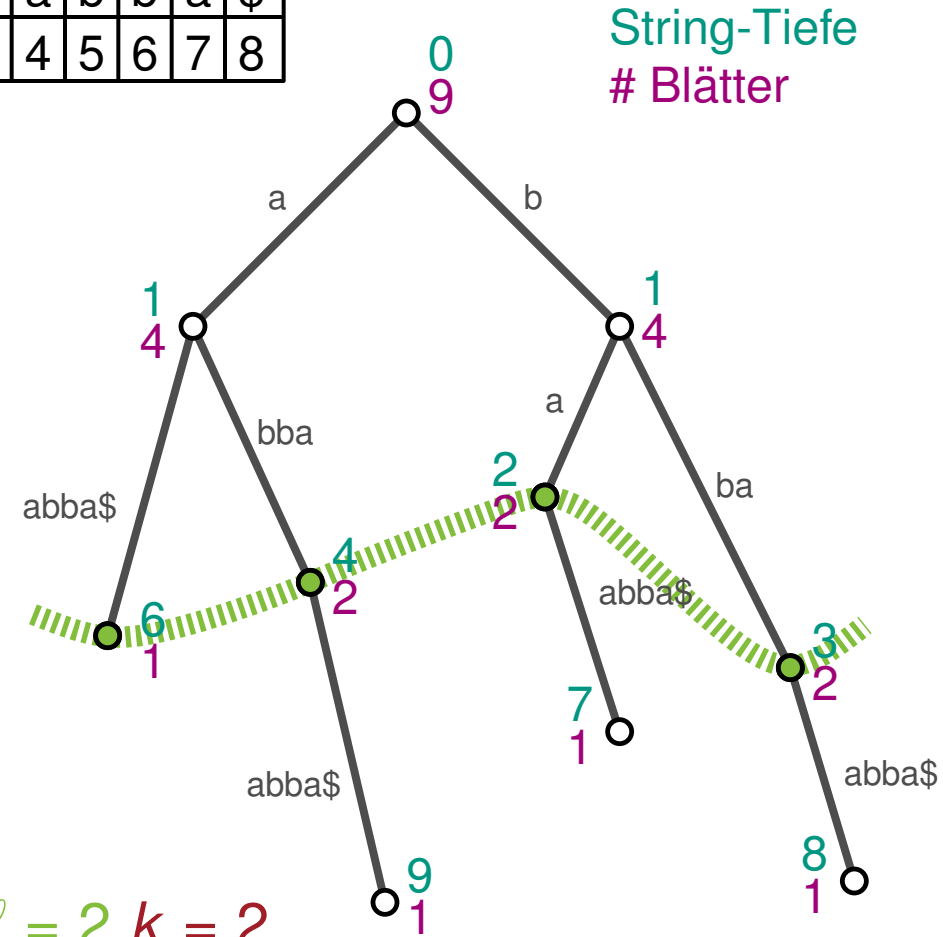
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8

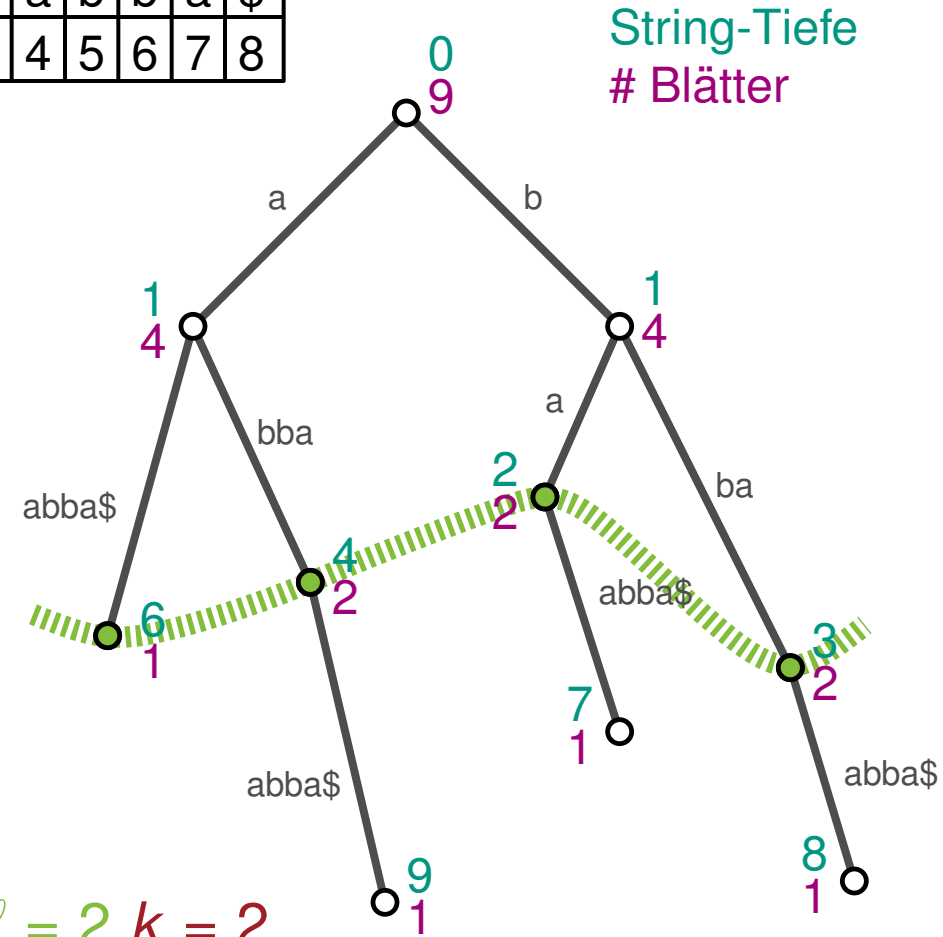




# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



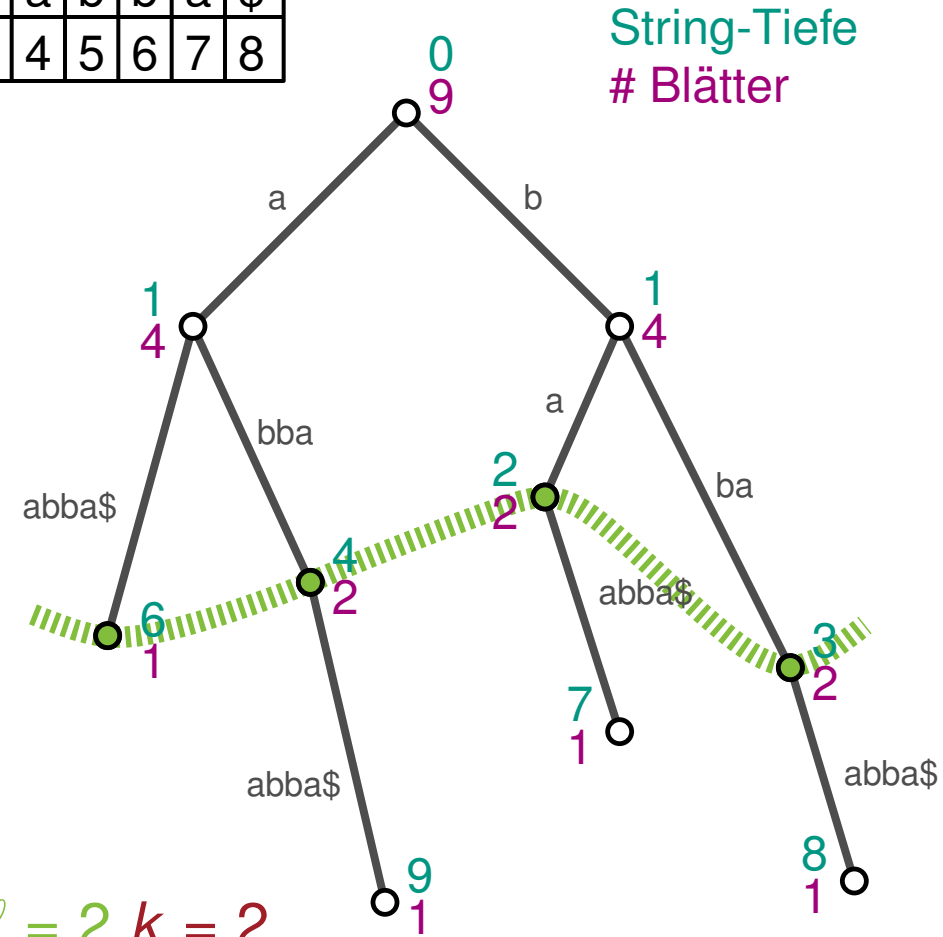
Kandidaten: 

3	0	2	1
1	2	2	2

# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



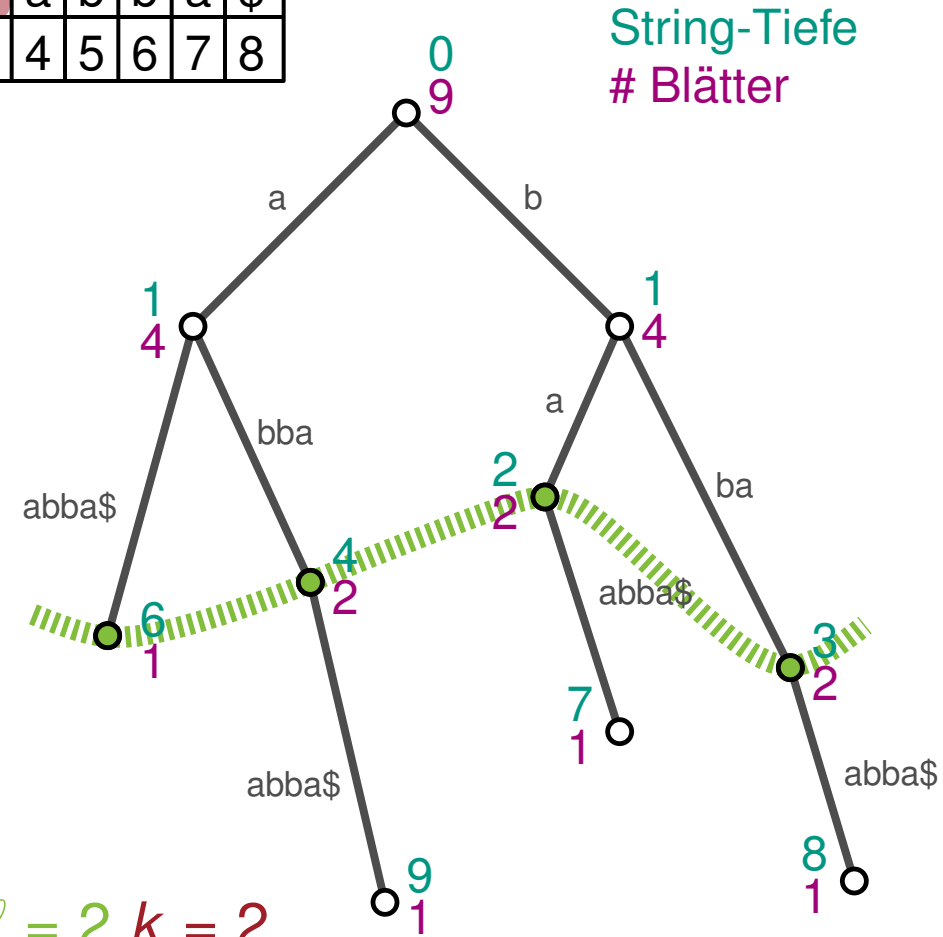
Query:  $\ell = 2$   $k = 2$

Kandidaten:  $\begin{matrix} 3 & 0 & 2 & 1 \\ 1 & 2 & 2 & 2 \end{matrix} \rightarrow \begin{matrix} 0 & 2 & 1 & 3 \\ 2 & 2 & 2 & 1 \end{matrix}$

# TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm  
Entferne dabei Sentinel-Blätter
- Für Query mit Eingaben  $\ell$  und  $k$ :
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe  $\geq \ell$   
→ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe  $k$ -ten Kandidaten aus.

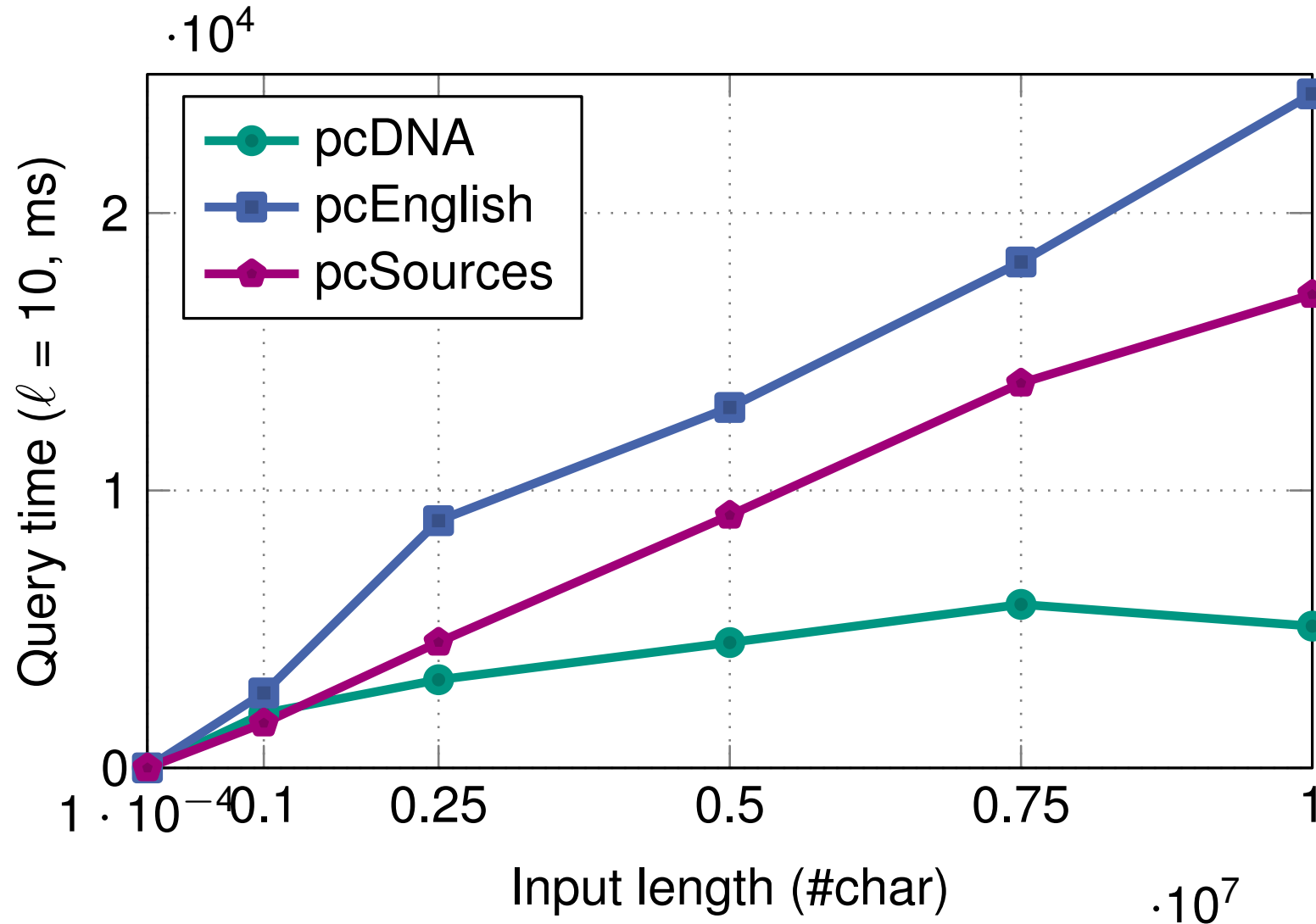
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



Query:  $\ell = 2$   $k = 2$

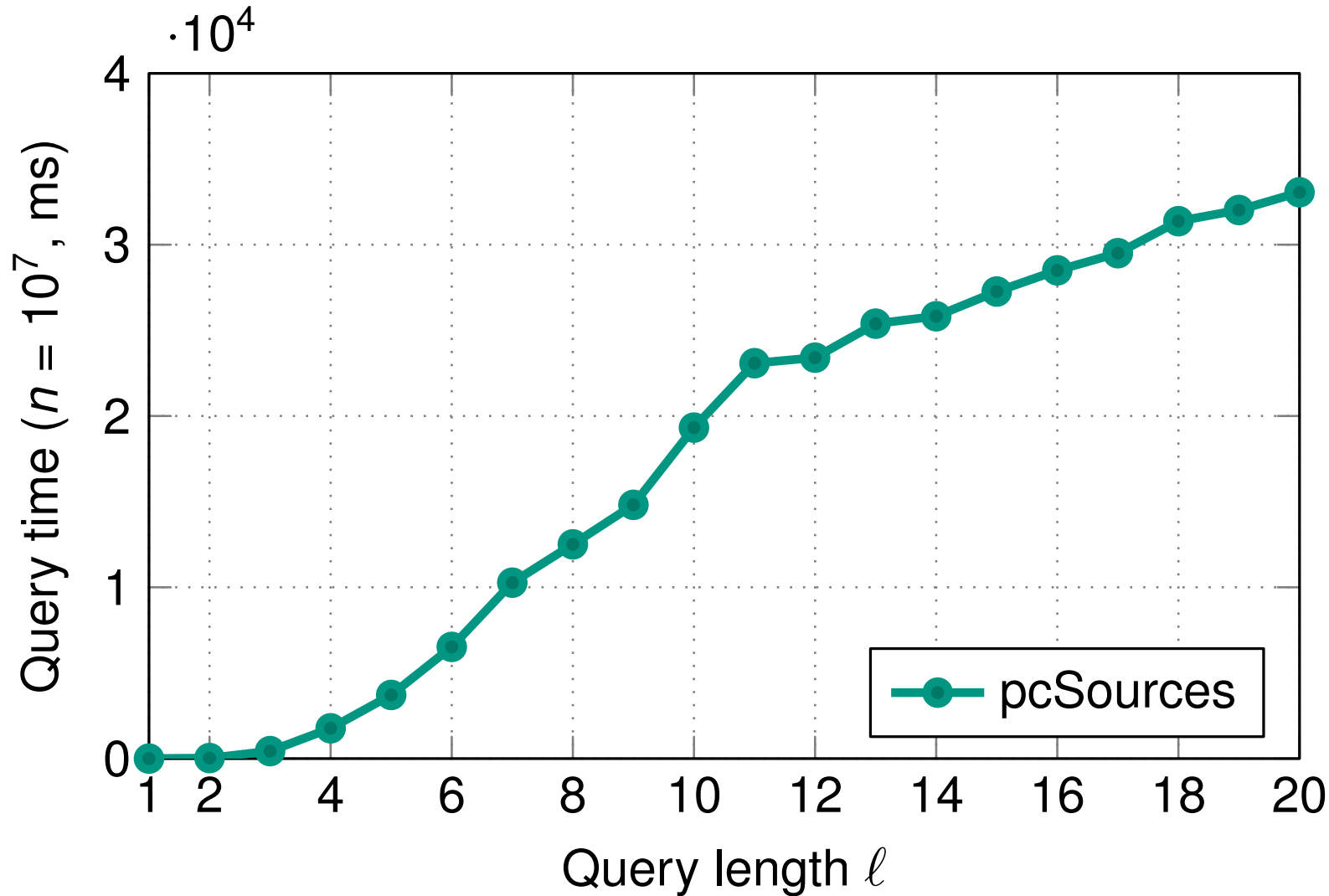
Kandidaten:  $\begin{matrix} 3 & 0 & 2 & 1 \\ 1 & 2 & 2 & 2 \end{matrix} \rightarrow \begin{matrix} 0 & 2 & 1 & 3 \\ 2 & 2 & 2 & 1 \end{matrix}$

# TopK Queries Performance



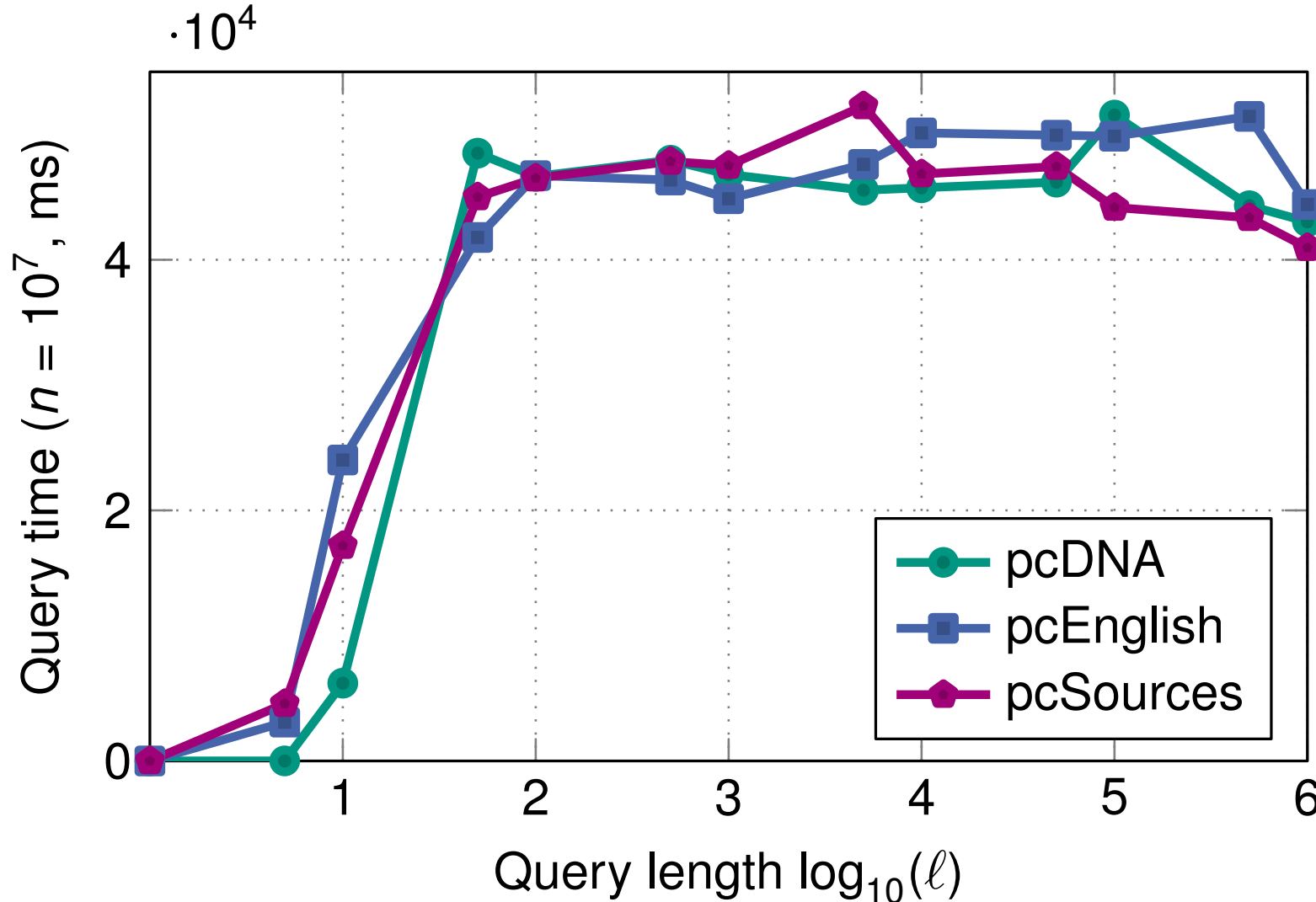
- Anfragezeit linear in Inputlänge
- Je repetetiver der Text, desto schneller die Anfrage: Weniger innere Knoten werden exploriert

# TopK Query Performance nach Patternlänge



- Anfragezeiten wachsen für kleine Patternlängen...
- ... bis Plateau erreicht wird.
- Mit höherer Länge müssen mehr innere Knoten eingesammelt werden,
- Aber irgendwann lässt der Effekt nach, man hat schon fast alle

# TopK Query Performance nach Patternlänge

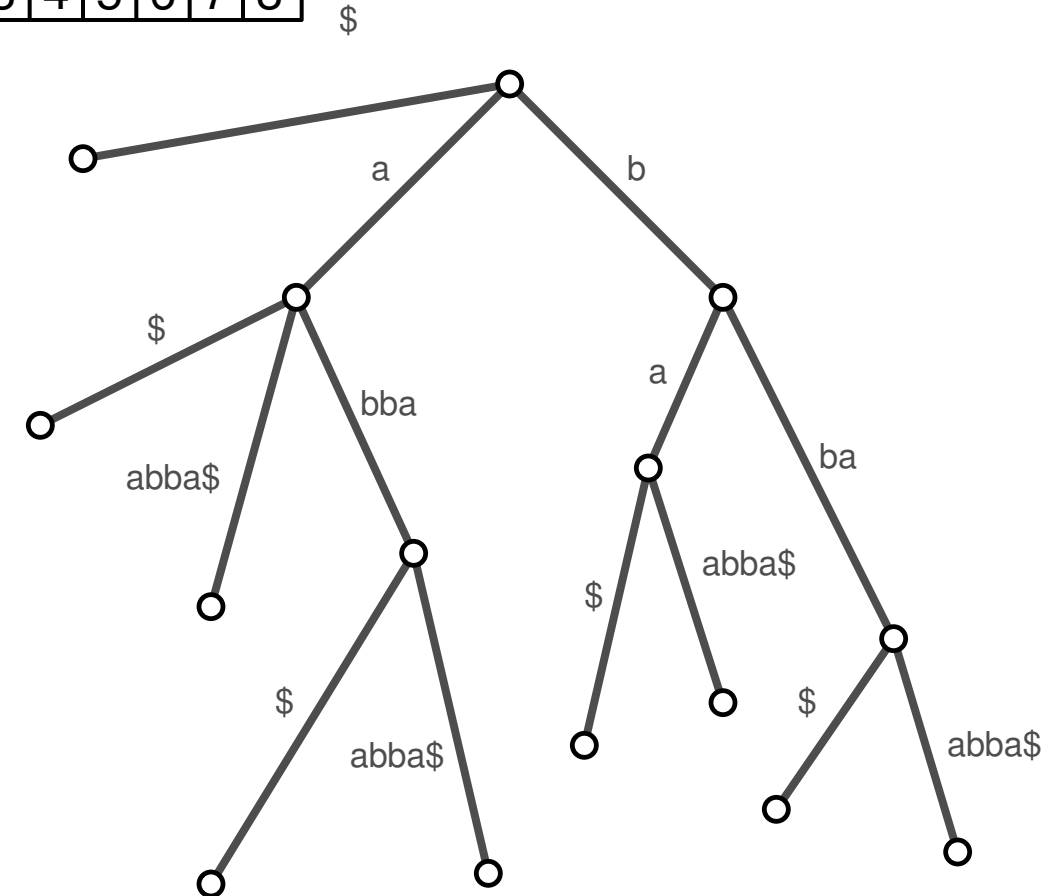


- Anfragezeiten wachsen für kleine Patternlängen...
- ... bis Plateau erreicht wird.
- Mit höherer Länge müssen mehr innere Knoten eingesammelt werden,
- Aber irgendwann lässt der Effekt nach, man hat schon fast alle

# Repeat Queries

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8

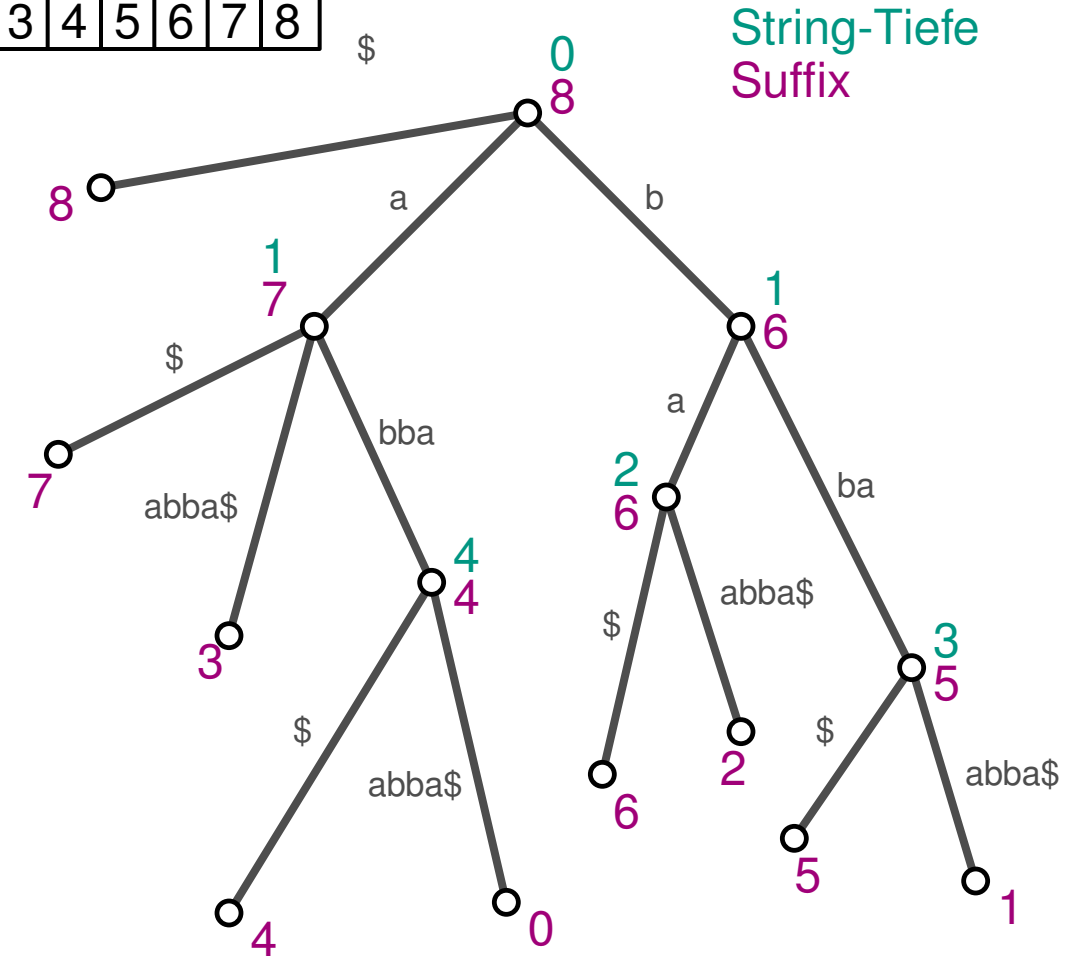
- Annotiere jeden Knoten mit String-Tiefe und repräsentiertem Suffix
- Sammle innere Knoten, absteigend sortiert nach String-Tiefe
- Für jeden inneren Knoten  $v$ :
  - Sammle sortierte Liste aller Suffixe unter dem Knoten
  - Falls es zwei Suffixe gibt mit Differenz der String-Tiefe von  $v$ : Gebe Ergebnis aus



# Repeat Queries

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8

- Annotiere jeden Knoten mit String-Tiefe und repräsentiertem Suffix
- Sammle innere Knoten, absteigend sortiert nach String-Tiefe
- Für jeden inneren Knoten  $v$ :
  - Sammle sortierte Liste aller Suffixe unter dem Knoten
  - Falls es zwei Suffixe gibt mit Differenz der String-Tiefe von  $v$ : Gebe Ergebnis aus

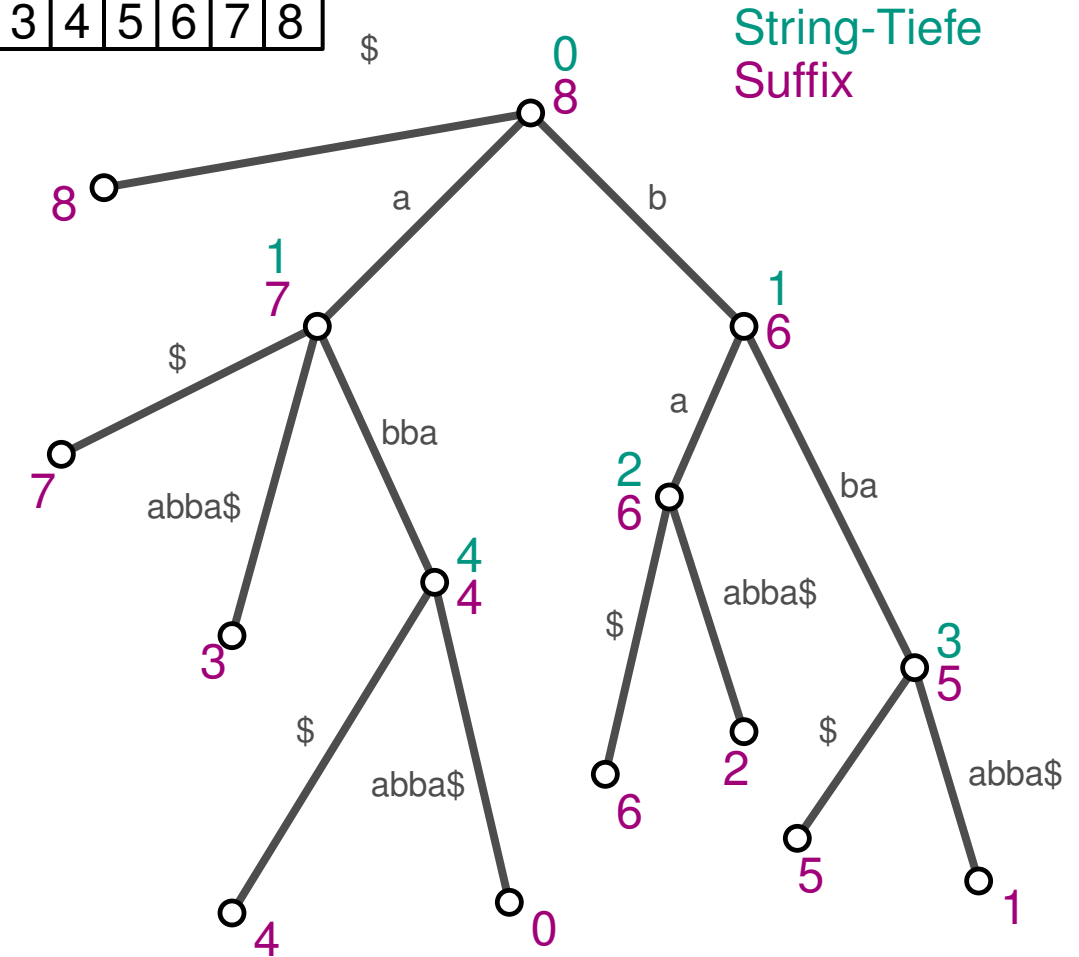




# Repeat Queries

- Annotiere jeden Knoten mit String-Tiefe und repräsentiertem Suffix
- Sammle innere Knoten, absteigend sortiert nach String-Tiefe
- Für jeden inneren Knoten  $v$ :
  - Sammle sortierte Liste aller Suffixe unter dem Knoten
  - Falls es zwei Suffixe gibt mit Differenz der String-Tiefe von  $v$ : Gebe Ergebnis aus

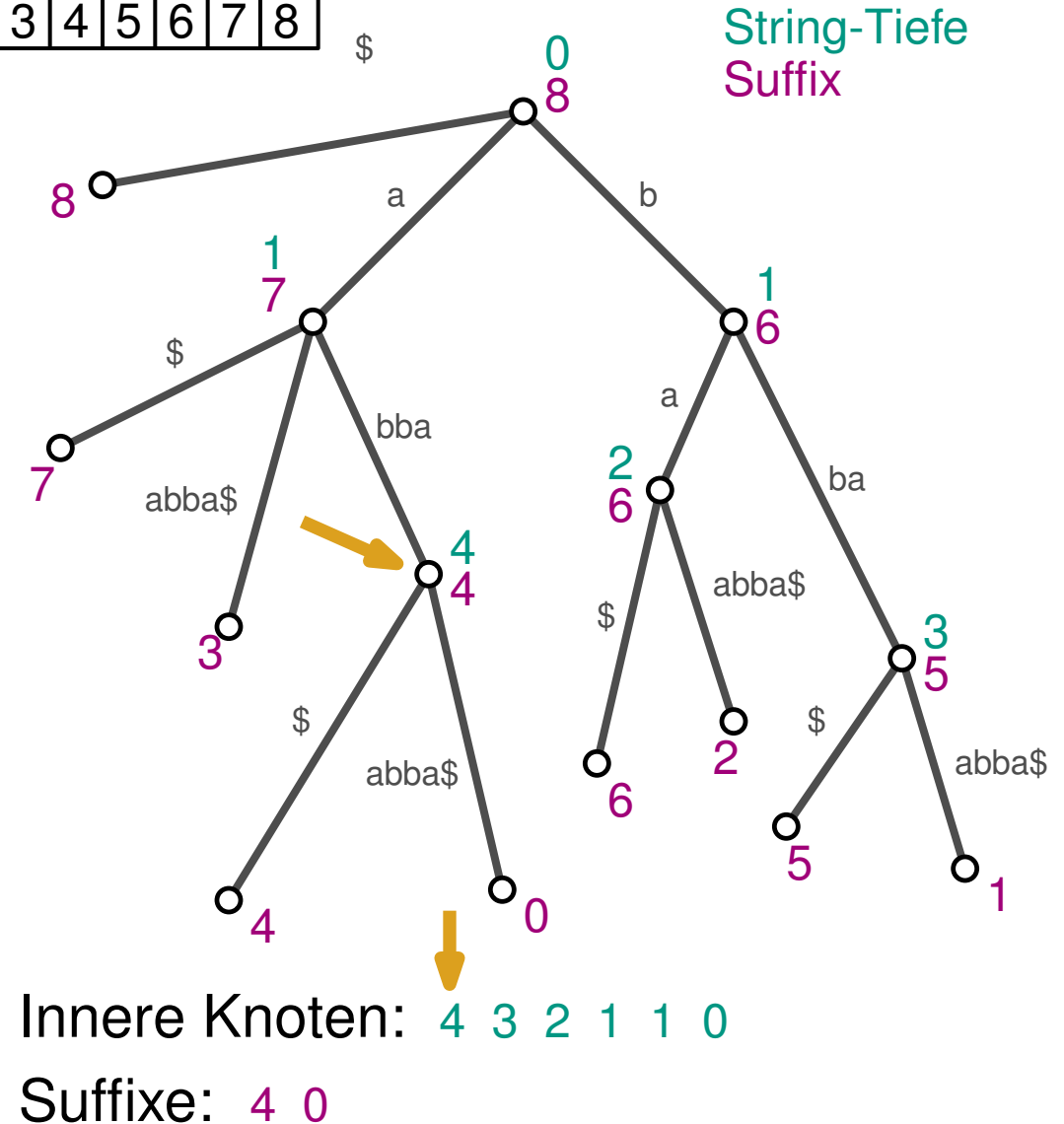
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# Repeat Queries

- Annotiere jeden Knoten mit String-Tiefe und repräsentiertem Suffix
- Sammle innere Knoten, absteigend sortiert nach String-Tiefe
- Für jeden inneren Knoten  $v$ :
  - Sammle sortierte Liste aller Suffixe unter dem Knoten
  - Falls es zwei Suffixe gibt mit Differenz der String-Tiefe von  $v$ : Gebe Ergebnis aus

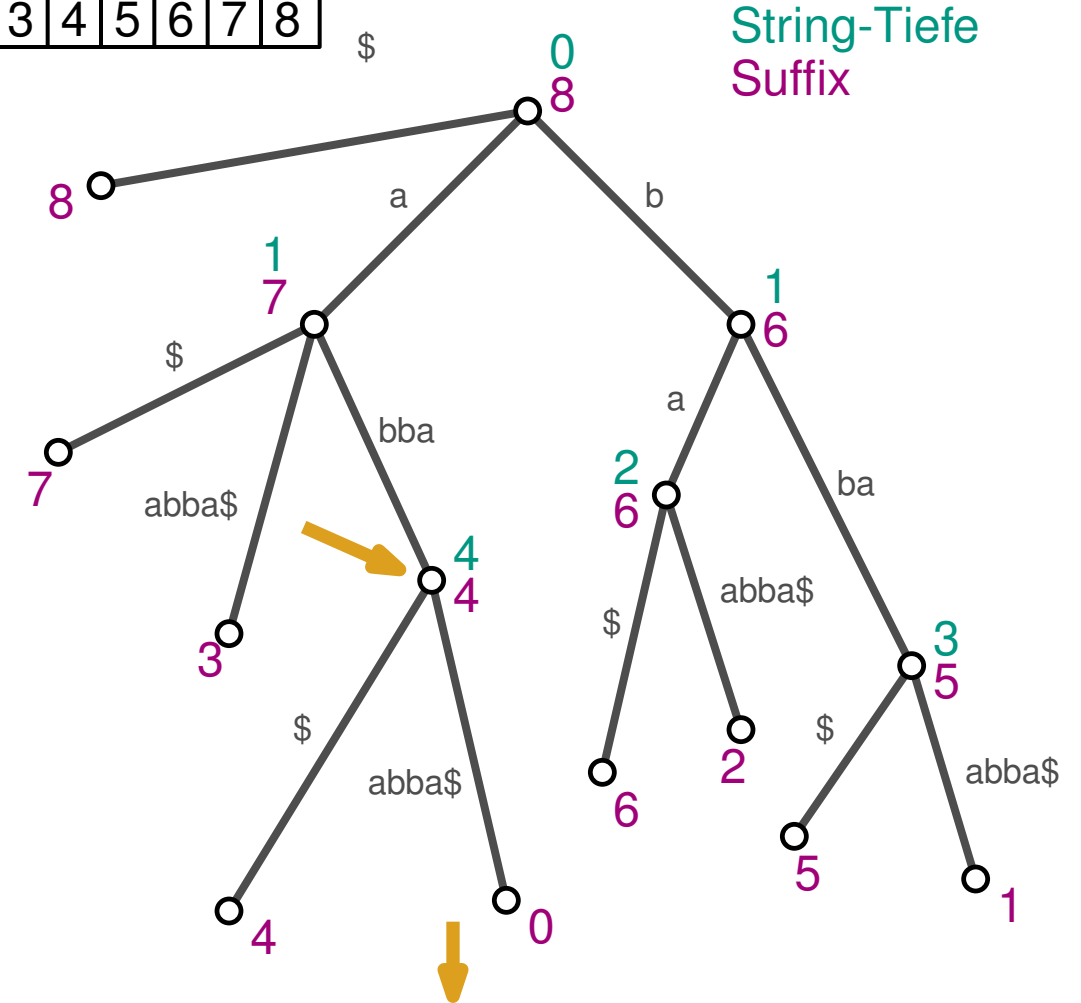
a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



# Repeat Queries

- Annotiere jeden Knoten mit String-Tiefe und repräsentiertem Suffix
- Sammle innere Knoten, absteigend sortiert nach String-Tiefe
- Für jeden inneren Knoten  $v$ :
  - Sammle sortierte Liste aller Suffixe unter dem Knoten
  - Falls es zwei Suffixe gibt mit Differenz der String-Tiefe von  $v$ : Gebe Ergebnis aus

a	b	b	a	a	b	b	a	\$
0	1	2	3	4	5	6	7	8



Innere Knoten: 4 3 2 1 1 0

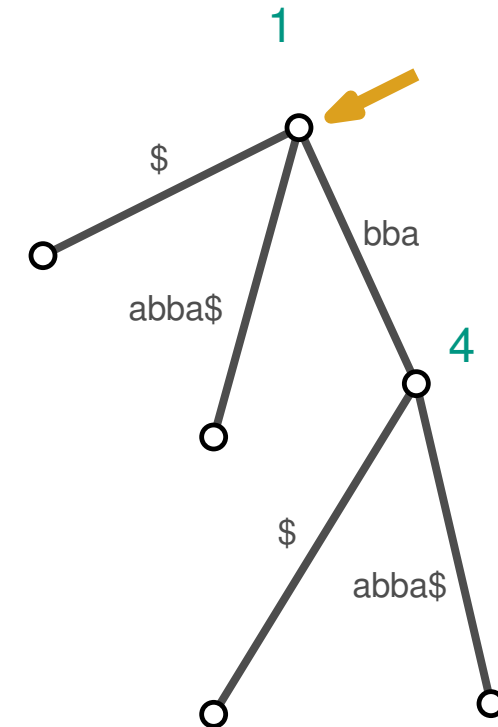
Suffixe: 4 0  $\rightarrow$  Ergebnis: (0, 8)

# Suffixe einsammeln als DP

**Ziel:** Berechne sortierte Liste aller Suffixe unter einem Knoten → Merge dazu alle Kinderlisten

- Es gibt zwei Fälle:
  - Kind ist Blatt → triviale einelementige Liste
  - Kind ist innerer Knoten.
    - **Beobachtung:** Alle Kinder haben höhere String-Tiefe  
→ Liste schon berechnet
- Merge die vorberechneten sortierten Listen aller Kinder
- Implementiert als DP.

String-Tiefe  
Suffix-Liste

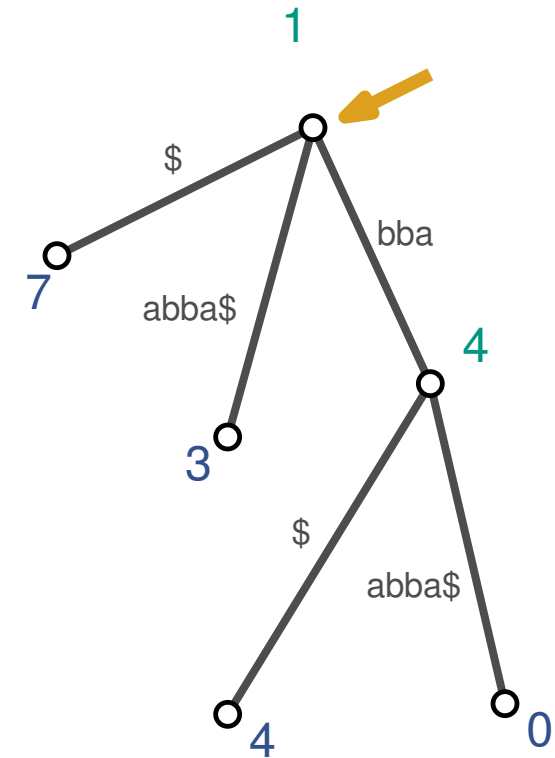


# Suffixe einsammeln als DP

**Ziel:** Berechne sortierte Liste aller Suffixe unter einem Knoten → Merge dazu alle Kinderlisten

- Es gibt zwei Fälle:
  - Kind ist Blatt → triviale einelementige Liste
  - Kind ist innerer Knoten.
    - **Beobachtung:** Alle Kinder haben höhere String-Tiefe  
→ Liste schon berechnet
- Merge die vorberechneten sortierten Listen aller Kinder
- Implementiert als DP.

String-Tiefe  
Suffix-Liste

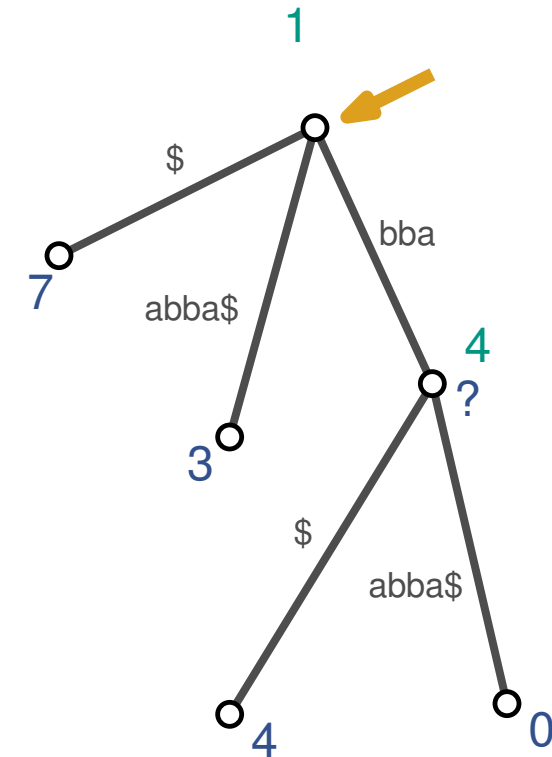


# Suffixe einsammeln als DP

**Ziel:** Berechne sortierte Liste aller Suffixe unter einem Knoten → Merge dazu alle Kinderlisten

- Es gibt zwei Fälle:
  - Kind ist Blatt → triviale einelementige Liste
  - Kind ist innerer Knoten.
    - **Beobachtung:** Alle Kinder haben höhere String-Tiefe  
→ Liste schon berechnet
- Merge die vorberechneten sortierten Listen aller Kinder
- Implementiert als DP.

String-Tiefe  
Suffix-Liste

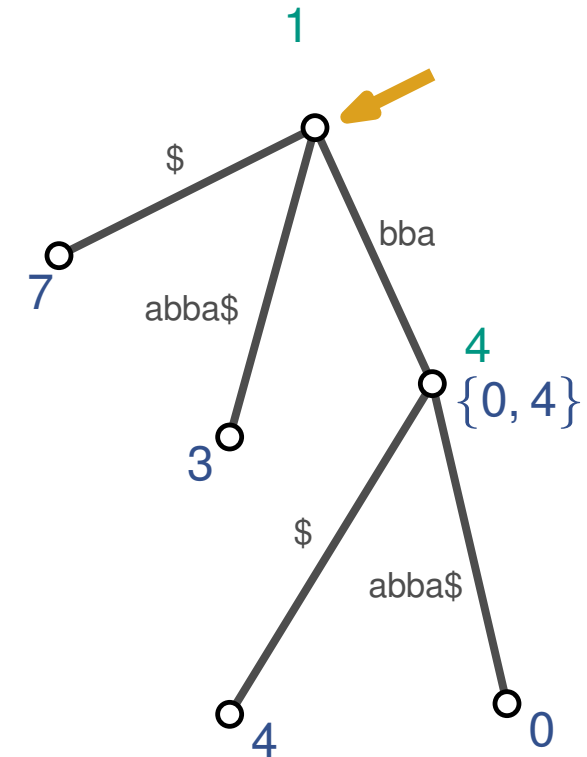


# Suffixe einsammeln als DP

**Ziel:** Berechne sortierte Liste aller Suffixe unter einem Knoten → Merge dazu alle Kinderlisten

- Es gibt zwei Fälle:
  - Kind ist Blatt → triviale einelementige Liste
  - Kind ist innerer Knoten.
    - **Beobachtung:** Alle Kinder haben höhere String-Tiefe  
→ Liste schon berechnet
- Merge die vorberechneten sortierten Listen aller Kinder
- Implementiert als DP.

String-Tiefe  
Suffix-Liste

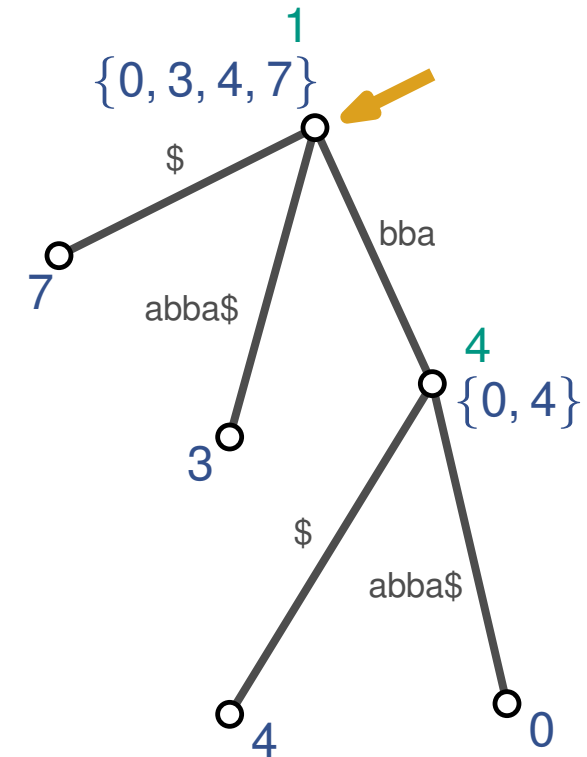


# Suffixe einsammeln als DP

**Ziel:** Berechne sortierte Liste aller Suffixe unter einem Knoten → Merge dazu alle Kinderlisten

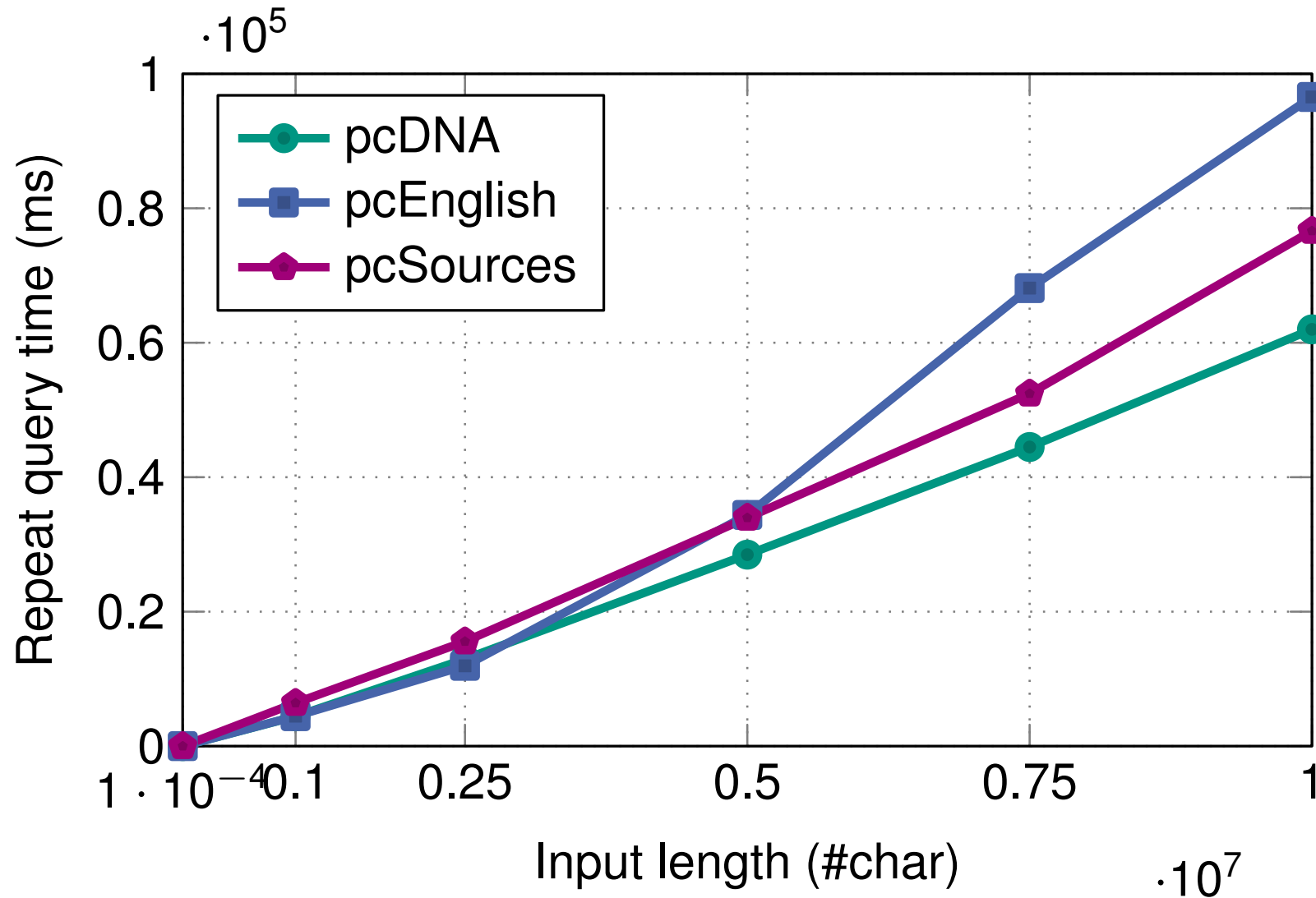
- Es gibt zwei Fälle:
  - Kind ist Blatt → triviale einelementige Liste
  - Kind ist innerer Knoten.
    - **Beobachtung:** Alle Kinder haben höhere String-Tiefe  
→ Liste schon berechnet
- Merge die vorberechneten sortierten Listen aller Kinder
- Implementiert als DP.

String-Tiefe  
Suffix-Liste





# Repeat Query Performance



- Query-Performance wieder linear in Eingabelänge
- Wieder: repetitivere Texte einfacher

