Textindexierung Programmierprojekt

Abschlusspräsentation · 07.02.2022
Moritz Potthoff

Vorberechnung: Suffix Tree

Suffix Tree-Konstruktion:

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

Figure with running times for different tex

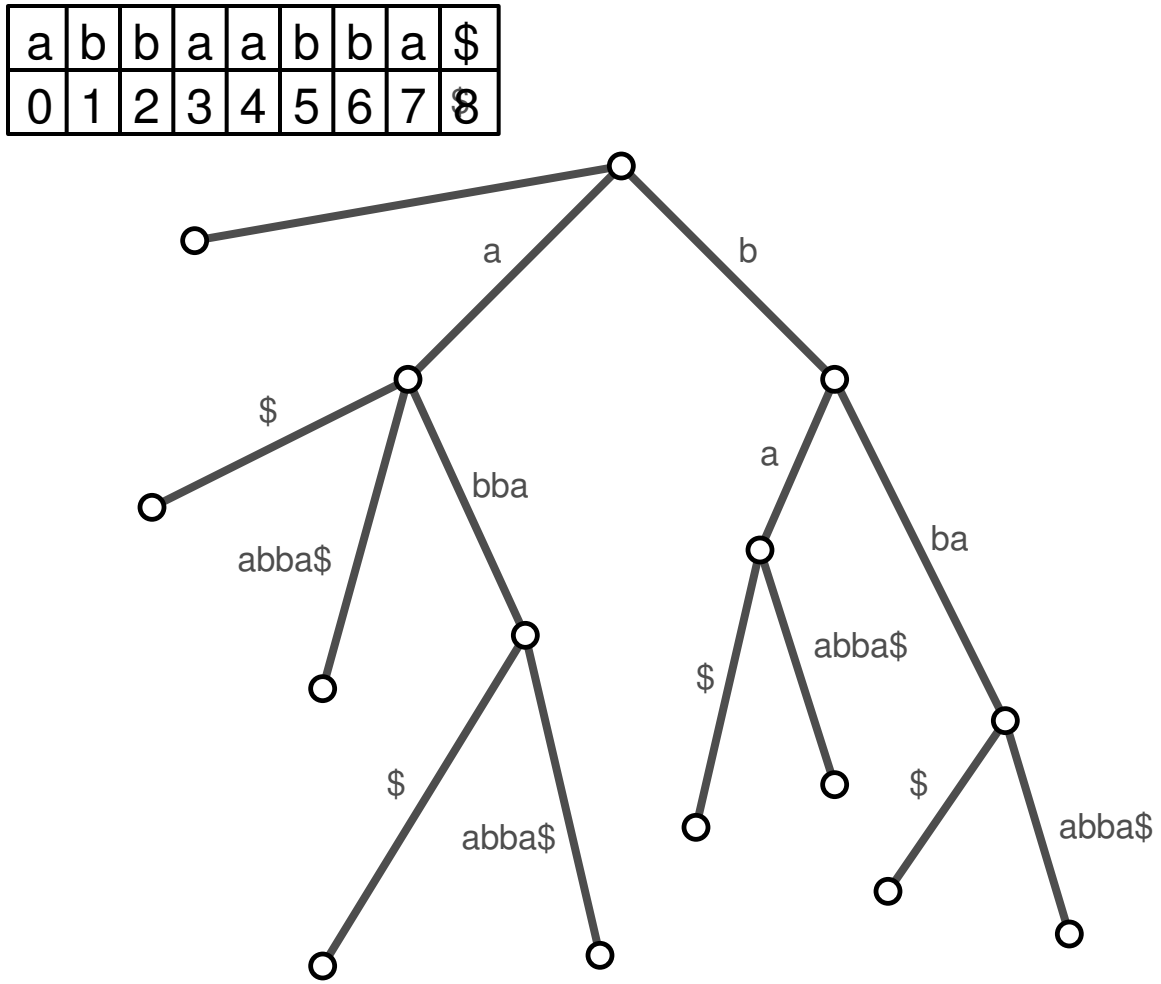
TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe k-ten Kandidaten aus.



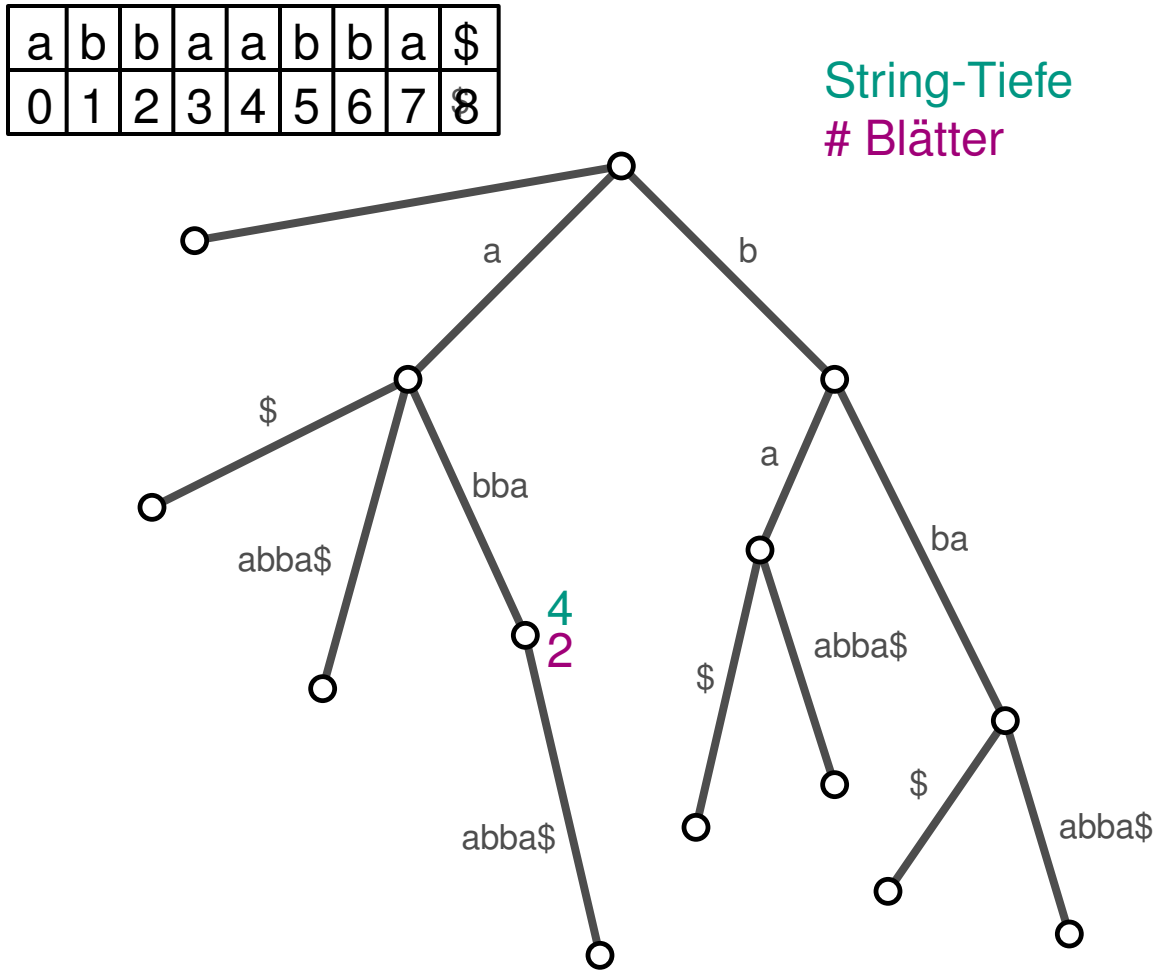
TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe k-ten Kandidaten aus.



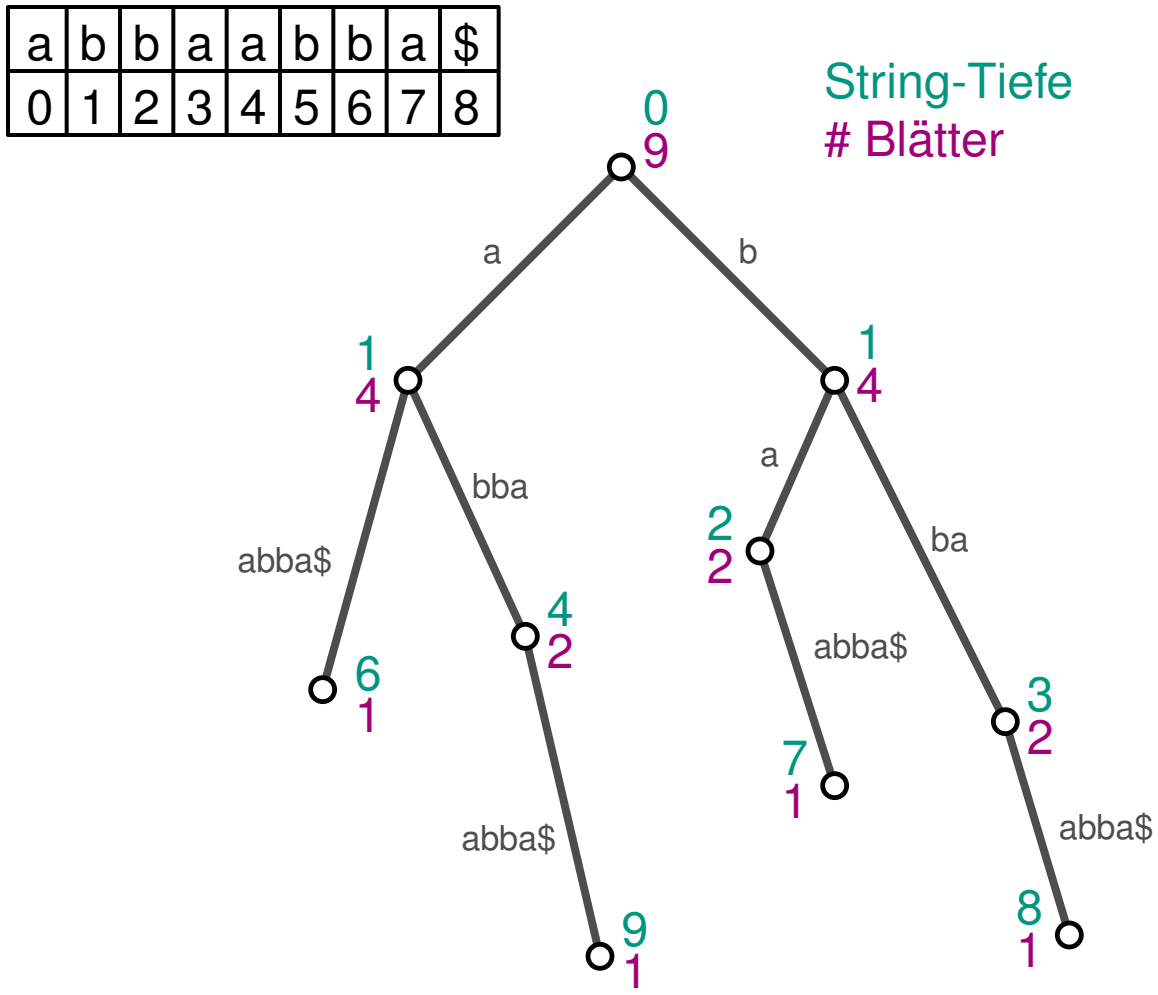
TopK-Queries – Algorithmus

- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **Kandidaten**, speichere Suffixposition und Anzahl Blätter
  - Sortiere Kandidaten stabil nach Anzahl Blättern
  - Gebe k-ten Kandidaten aus.



TopK-Queries – Algorithmus

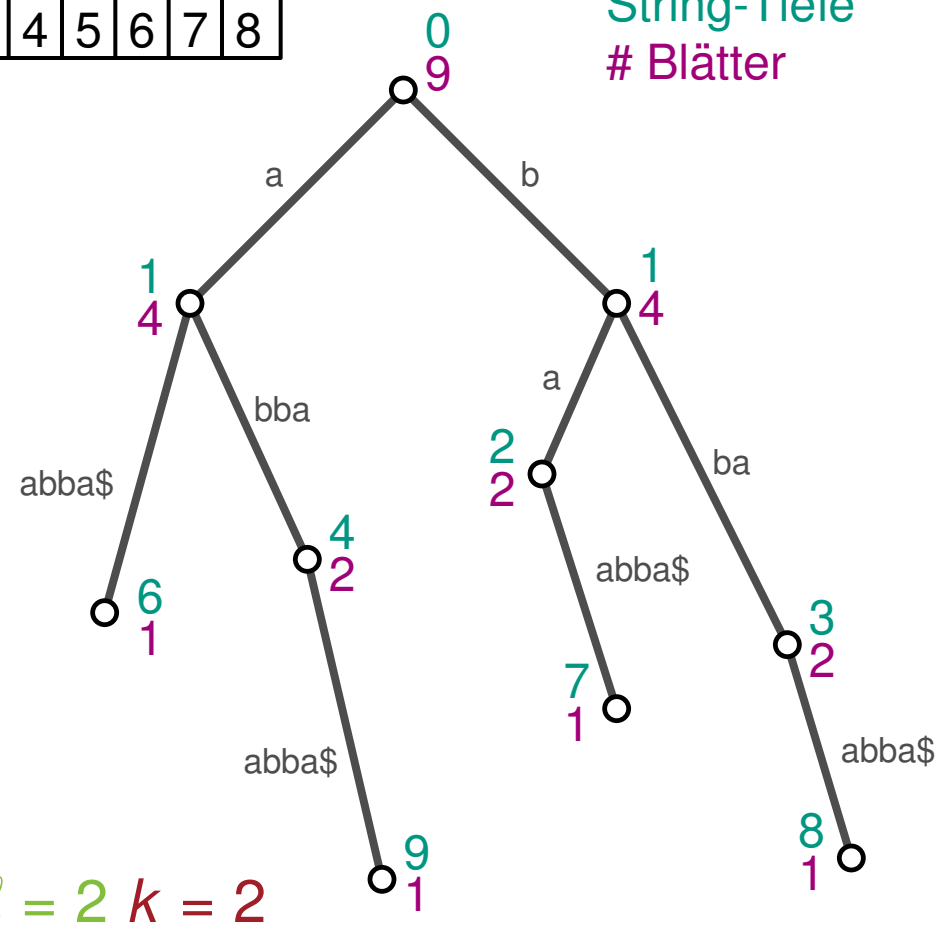
- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **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: $\ell = 2$ $k = 2$

TopK-Queries – Algorithmus

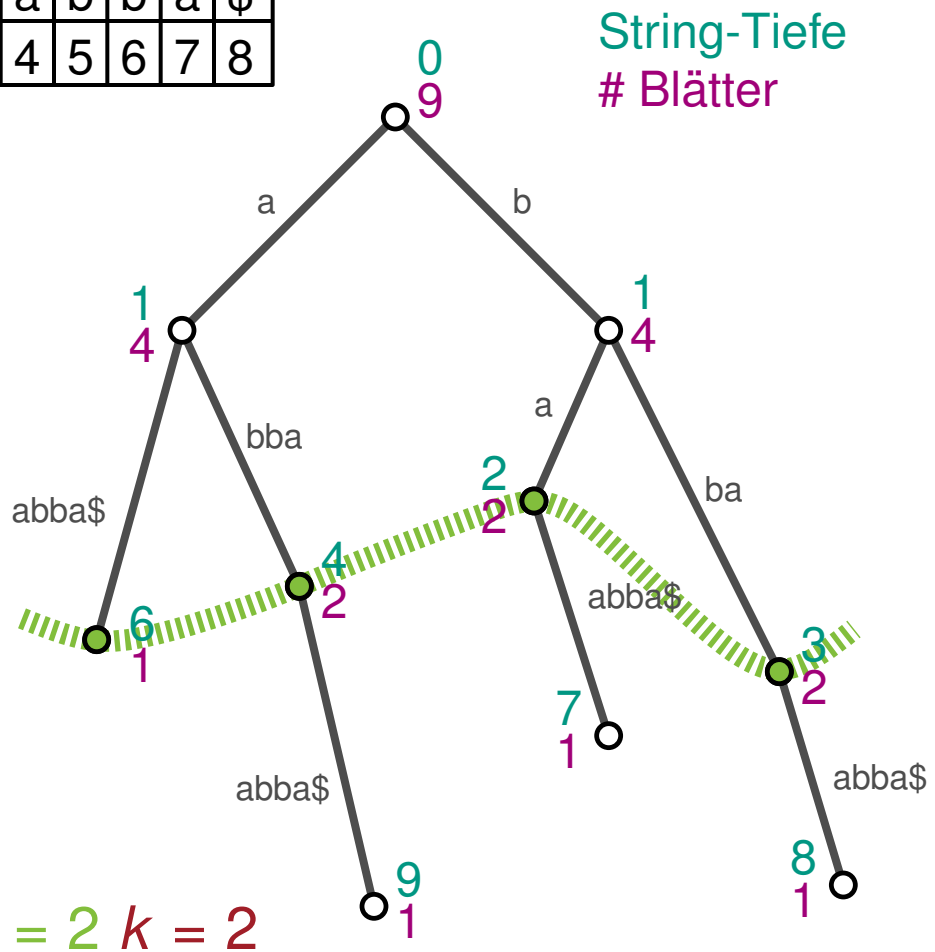
- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **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: $\ell = 2$ $k = 2$

TopK-Queries – Algorithmus

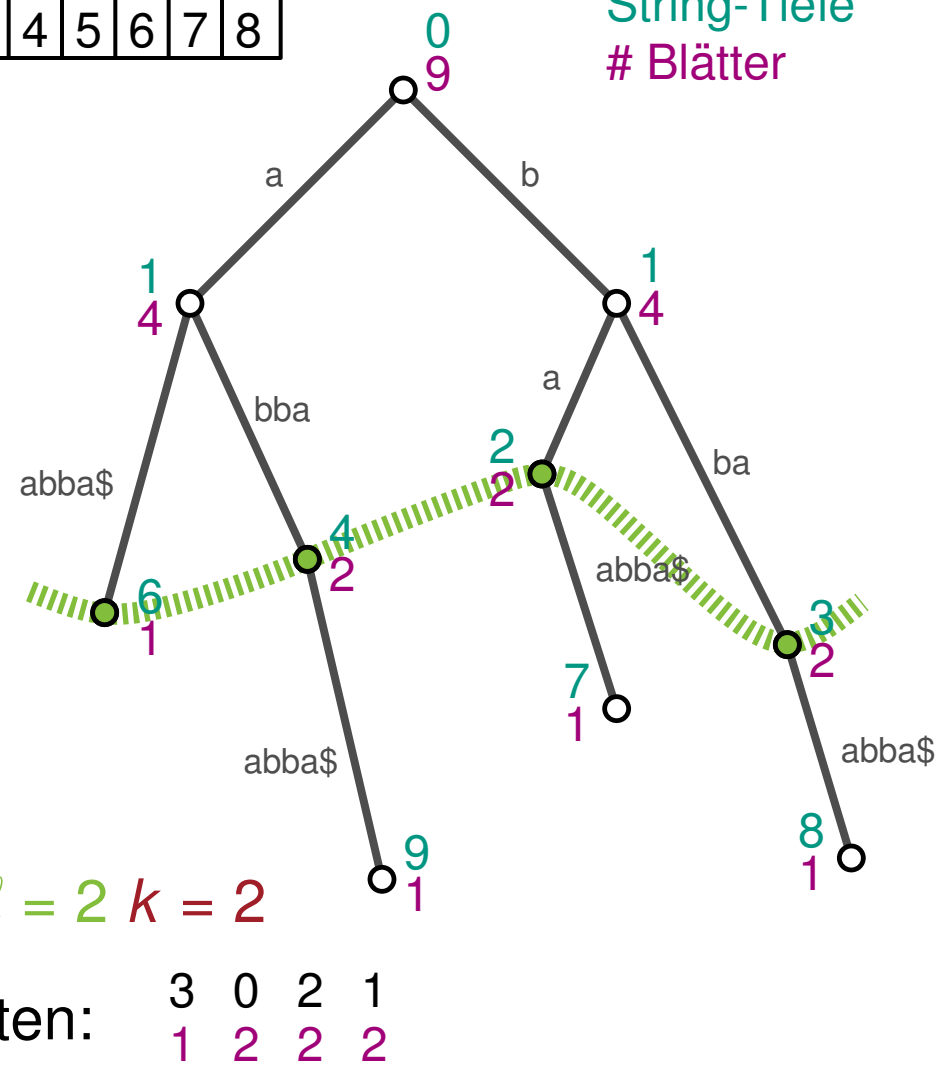
- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **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:

3	0	2	1
1	2	2	2

TopK-Queries – Algorithmus

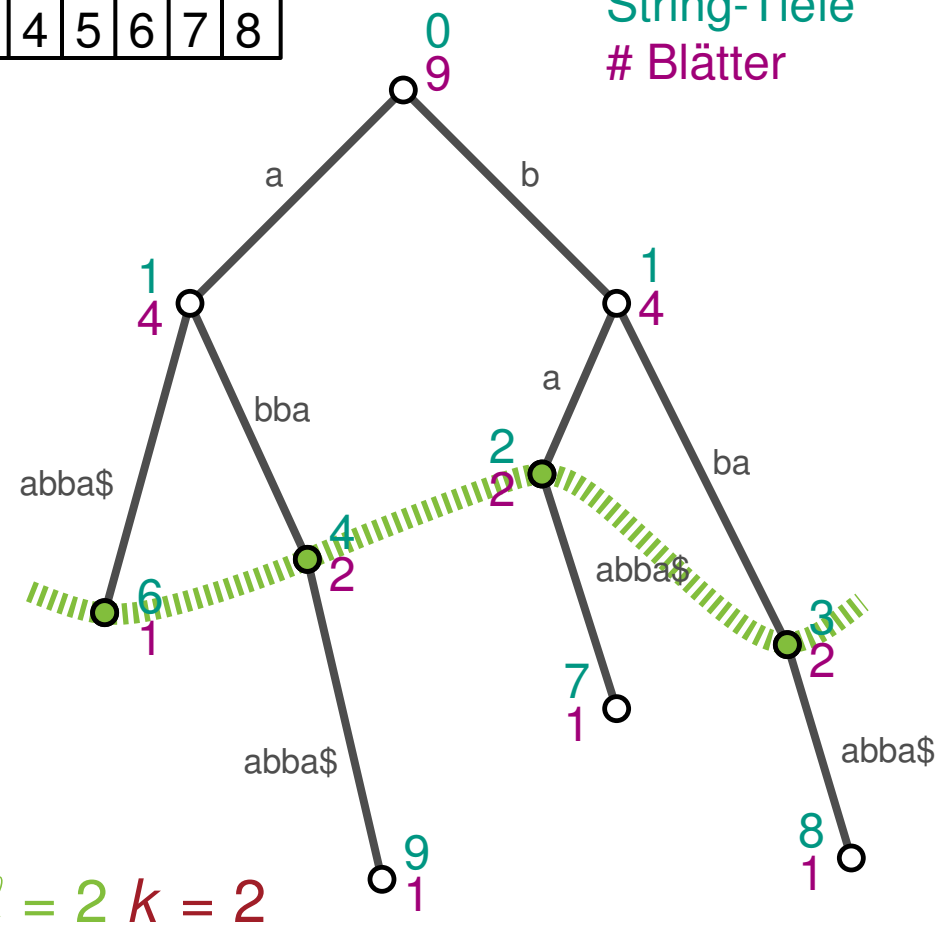
- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **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:

3	0	2	1	$\rightarrow$	0	2	1	3
1	2	2	2		2	2	2	1

TopK-Queries – Algorithmus

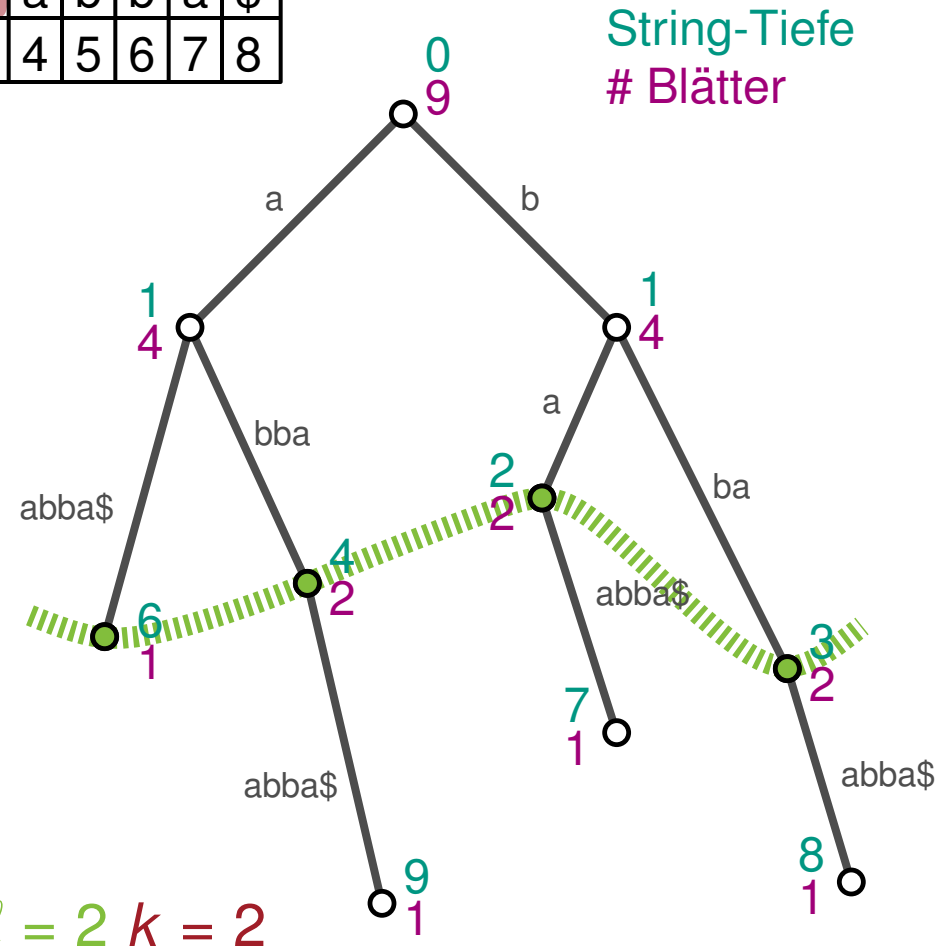
- Annotiere jeden Knoten mit String-Tiefe und Anzahl Blätter unter ihm

  Entferne dabei Sentinel-Blätter

- Für Query mit Eingaben ℓ und k:
  - DFS: Sammle jeden höchsten Knoten mit String-Tiefe $\geq \ell$

    $\rightarrow$ **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: $\ell = 2$ $k = 2$

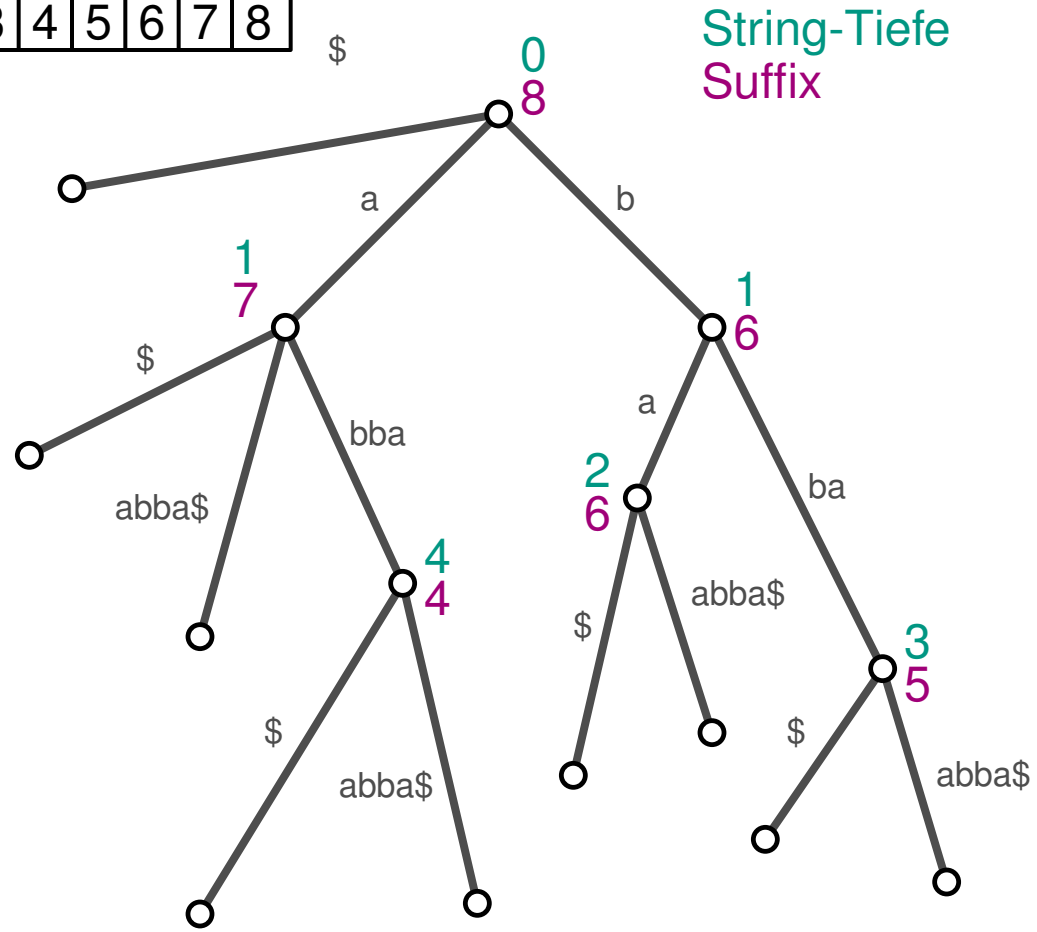
Kandidaten:

3	0	2	1	$\rightarrow$	0	2	1	3
1	2	2	2		2	2	2	1

TopK Queries – Evaluation

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 möglichem 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