

# Text Indexing

## Lecture 10: Top- $k$ Document Retrieval

Florian Kurpicz

The slides are licensed under a Creative Commons Attribution-ShareAlike 4.0 International License © ⓘ ⓘ: [www.creativecommons.org/licenses/by-sa/4.0](https://www.creativecommons.org/licenses/by-sa/4.0) | commit 59da60d compiled at 2025-01-13-13:01



<https://pingo.scc.kit.edu/815630>

# Document Listing

- similar to document retrieval
- get all documents containing a *phrase*

## Definition: Document Listing

Given a collection of  $D$  documents  $\mathcal{D} = \{d_1, d_2, \dots, d_D\}$  containing symbols from an alphabet  $\Sigma = [1, \sigma]$  and a pattern  $P \in \Sigma^*$ , return all  $j \in [1, D]$ , such that  $d_j$  contains  $P$ .

- $d_1 = \text{ATA}$
- $d_2 = \text{TAAA}$
- $d_3 = \text{TATA}$

And for queries:

- $P = \text{TA}$  is contained in  $d_1, d_2$ , and  $d_3$
- $P = \text{ATA}$  is contained in  $d_1$  and  $d_3$

# Basic Concepts

## Definition: Document Concatenation

Given a collection of  $D$  documents  $\mathcal{D} = \{d_1, d_2, \dots, d_D\}$  containing symbols from an alphabet  $\Sigma = [1, \sigma]$  where each document **ends with a special symbol**  $\# \notin \Sigma$ , the string

$$\mathcal{C} = d_1 d_2 \dots d_D \$$$

is called the concatenation of the documents with  $\$ \notin \Sigma$  and  $\$ < \# < \alpha$  for all  $\alpha \in \Sigma$

- $N = |\mathcal{C}| = \sum_{i=1}^D |d_i|$

- $d_1 = \text{ATA}$
- $d_2 = \text{TAAA}$
- $d_3 = \text{TATA}$

Document Concatenation:

- $\text{ATA}\#\text{TAAA}\#\text{TATA}\#\text{\$}$

# Suffix Array for Document Concatenation

- given a document concatenation  $\mathcal{C}$ , build the suffix array
- requires  $O(n)$  time
- entries in suffix array correspond to documents


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T$	A	T	A	#	T	A	A	A	#	T	A	T	A	#	\$
SA	15	14	4	9	13	3	8	7	6	11	1	12	2	5	10
DA	0	3	1	2	3	1	2	2	2	3	1	3	1	2	3

## Definition: Document Array

Given a document concatenation  $\mathcal{C}$  and its suffix array  $SA$ , the **document array**  $DA$  is defined as

$$DA[i] = j \iff \sum_{k=1}^{j-1} |d_k| < SA[i] \leq \sum_{k=1}^j |d_k|$$

for  $i > 1$  and  $DA[1] = 0$

-  **PINGO** are the document and suffix array sufficient to return all documents?

# Naive Document Listing

- given document concatenation  $\mathcal{C}$ , its suffix array  $SA$ , and document array  $DA$
- enhance suffix array to do pattern matching in  $O(|P|)$  time ⓘ only briefly discussed in lecture
- find interval in suffix array matching  $P$
- report all documents in interval in  $DA$
- problem:  $O(|P| + N)$  query time ⓘ very bad

- is there a better solution?
- better query time
- better (or at least equal) space requirements?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T$	A	T	A	#	T	A	A	A	#	T	A	T	A	#	\$
$SA$	15	14	4	9	13	3	8	7	6	11	1	12	2	5	10
$DA$	0	3	1	2	3	1	2	2	2	3	1	3	1	2	3

- $P = TA$

# Optimal Time Document Listing (1/2) [Mut02]

## Definition: Chain Array


Given document concatenation  $\mathcal{C}$ , its suffix array  $SA$ , and document array  $DA$ , the **chain array**  $CA$  is defined as

$$CA[i] = \max\{j < i : DA[j] = DA[i]\} \cup \{0\}$$

- chains same documents together
- find lexicographically smaller suffix of same document
- use it to report documents just once
- build RMQ data structure for CA

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T$	A	T	A	#	T	A	A	A	#	T	A	T	A	#	\$
$SA$	15	14	4	9	13	3	8	7	6	11	1	12	2	5	10
$DA$	0	3	1	2	3	1	2	2	2	3	1	3	1	2	3
$CA$	0	0	0	0	2	3	4	7	8	5	6	10	11	9	12

- $P = TA$

-  **PINGO** is the chain array with RMQs enough to list all documents in optimal time?

## Optimal Time Document Listing (2/2)

- given document concatenation  $\mathcal{C}$ , its suffix array  $SA$ , document array  $DA$ , and chain array  $CA$  with RMQ data structure
- find interval  $SA[s, e]$  as before
- report document  $DA[m]$  only if  $CA[m] < s$  for  $m \in [s, e]$

- find all positions where  $CA[m] < s$  with RMQs
- get arg min of  $CA$  in interval and report  $DA[m]$  if  $CA[m] < s$
- split interval in  $[s, m - 1]$  and  $[m + 1, e]$  and recurse
- ignore intervals where nothing is reported

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T$	A	T	A	#	T	A	A	A	#	T	A	T	A	#	\$
$SA$	15	14	4	9	13	3	8	7	6	11	1	12	2	5	10
$DA$	0	3	1	2	3	1	2	2	2	3	1	3	1	2	3
$CA$	0	0	0	0	2	3	4	7	8	5	6	10	11	9	12

■  $P = TA$

### Lemma: Optimal Document Listing

Listing all documents containing a pattern  $P$  can be done in  $O(|P| + occ)$  time



# Top- $k$ Document Retrieval for Single-Term Frequencies

## Definition: Top- $k$ Document Retrieval

Given a collection of  $D$  documents  $\mathcal{D} = \{d_1, d_2, \dots, d_D\}$  containing symbols from an alphabet  $\Sigma = [1, \sigma]$ , a pattern  $P \in \Sigma^*$ , and a **threshold**  $k$ , return the top- $k$  documents  $j \in [1, D]$ , such that  $d_j$  contains  $P$  most often

- retrieve  $occ$  distinct documents where  $P$  occurs
- determine frequency of  $P$  in each document
- maintain min-heap of (frequency,document)-pairs of size  $k$
- total time:  $O(|P| + occ(\lg k + \lg N))$

- $occ$  can be  $N$
- can we do better

- optimal solution:  $O(|P| + k)$  query time in  $O(N \lg N)$  bits [NN12]
- now:  $O(|P| + k \lg N)$  [GKN17]

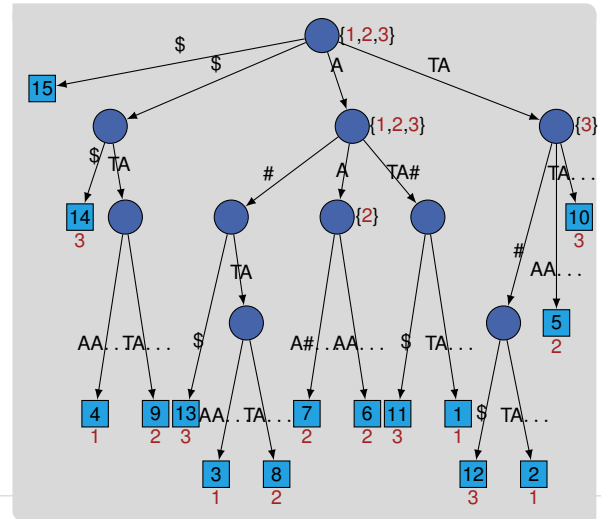


# Generalized Suffix Tree for Top- $k$ Document Retrieval (1/4)

- a **generalized** suffix tree is a suffix tree for a set of strings
- document concatenation is a set of strings

## Mark Document Numbers

- mark all leaves with  $DA$ -entry  $i$
- add  $i$  to nodes that are lowest common ancestor of two leaves marked with  $i$




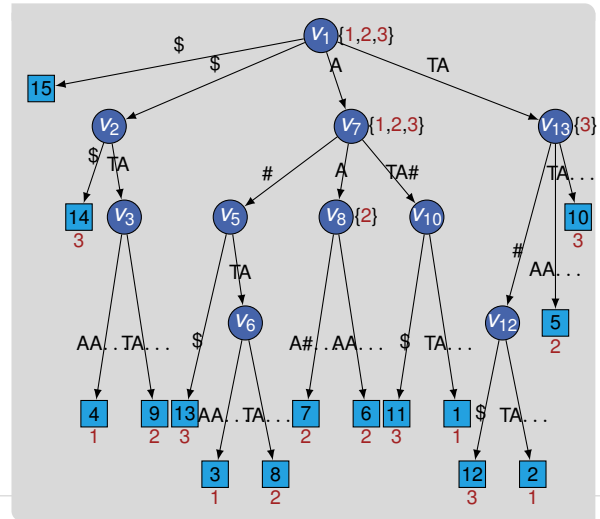
# Generalized Suffix Tree for Top- $k$ Document Retrieval (2/4)

## Inner Node Names

- leaf index is rank of suffix in  $[1, M]$  in leaf
- each inner node gets  $v$  gets  $id(v)$ , which is the leaf index of rightmost leaf in leftmost child

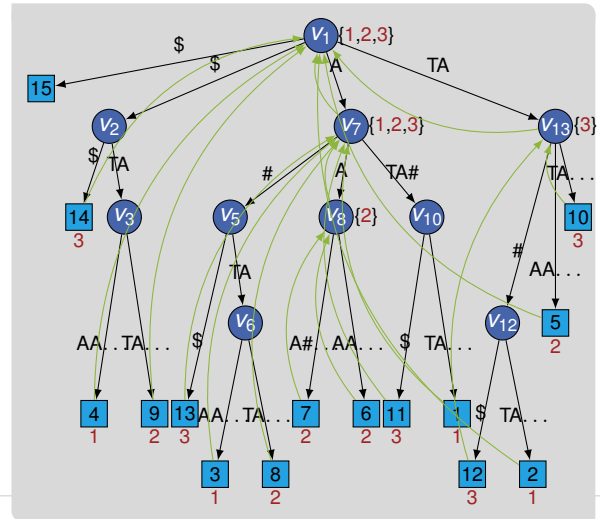
- $id(v) \neq id(w)$  for all inner nodes  $v \neq w$
- $id(v) \in [1, M]$
- $id(v) - 1 \in [lb(v), rb(v)]$ , with  $[lb(v), rb(v)]$  being  $v$ 's suffix array interval

- example on the board 



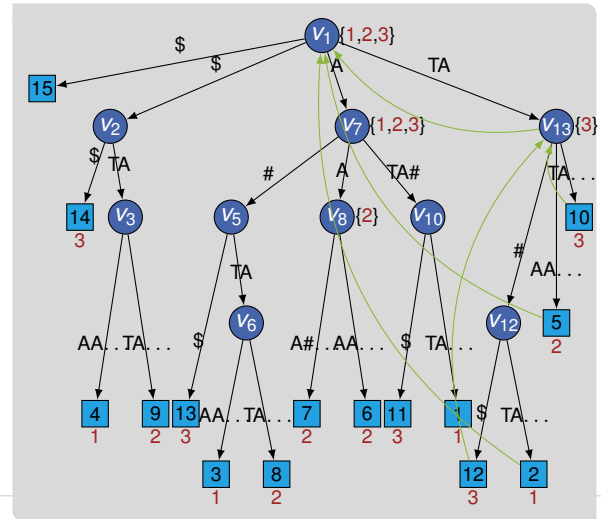
# Generalized Suffix Tree for Top- $k$ Document Retrieval (3/4)

- connect node with id  $i$  to closest ancestor containing id  $i$
- nodes marked with id  $i$  correspond to suffix tree of  $d_i$
- document id  $i$  occurs at most  $|d_i|$  times in leaves and  $|d_i| - 1$  times in inner nodes
- there are at most  $O(N)$  document ids in the generalized suffix tree





# Generalized Suffix Tree for Top- $k$ Document Retrieval (4/4)

- to retrieve documents containing pattern  $P$
  - select **locus** of  $P$   $\circledast$  first node  $v$  with  $P$  is prefix of  $\lambda(v)$
- per document at most one pointer leaves subtree of locus  $v$
  - associate each pointer with number of occurrences of documents in pointers source (weight)
  - pointer of document  $i$  leaving subtree has maximum weight of all document  $i$  pointers in subtree
  - document listing is listing all documents of pointers leaving subtree

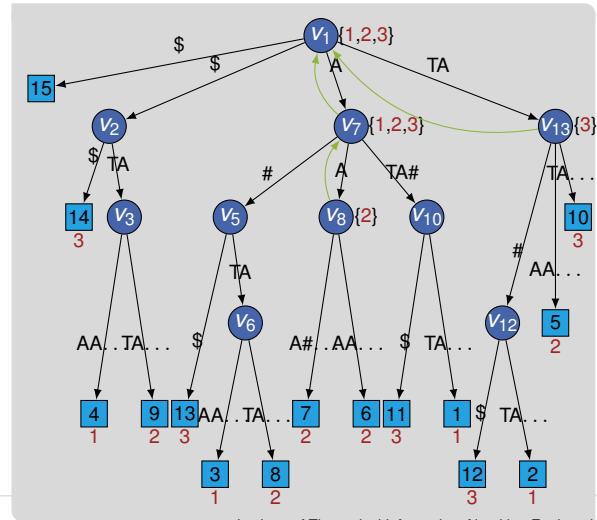


# Representing Pointers on a Grid (1/2)

- now: report top- $k$  documents
- represent pointers in a grid 
- for simplicity only weights  $\geq 2$   starting at inner node

- assign each pointer to  $(x, y)$ -coordinate
  - $x$ :  $id(\text{source})$
  - $y$ :  $d(\text{target})$
- each point is associated with pointers weight

- given a locus  $v$ , all pointers leaving the subtree have  $y$ -coordinate  $< d(v)$





## Representing Pointers on a Grid (2/2)

- grid can be represented using wavelet tree
- range **maximum** query for each level

### Answering Queries

- find string depth of locus in suffix tree
- answer range query in grid
- if represented as wavelet tree, use RMQs on each level to report top- $k$  documents
- if  $\leq k$  documents, use document listing
- total time:  $O(m + k \lg N)$

-  **PINGO** how can we represent the pointers in a grid?

- example range queries in wavelet trees on the board 

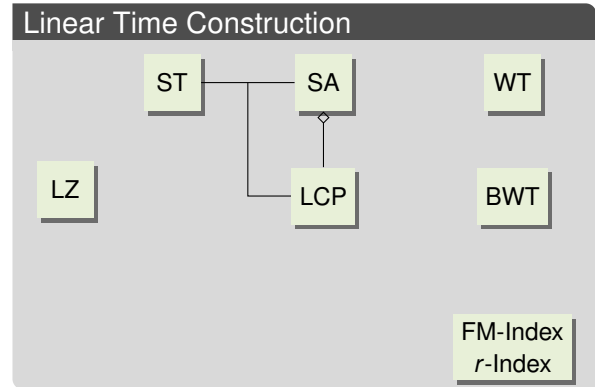


# Conclusion and Outlook

## This Lecture

- document listing
- top- $k$  document retrieval (single term frequency)

## Linear Time Construction



# Bibliography I

- [GKN17] Simon Gog, Roberto Konow, and Gonzalo Navarro. “Practical Compact Indexes for Top- $k$  Document Retrieval”. In: *ACM J. Exp. Algorithmics* 22 (2017). DOI: [10.1145/3043958](https://doi.org/10.1145/3043958).
- [Mut02] S. Muthukrishnan. “Efficient Algorithms for Document Retrieval Problems”. In: *SODA. ACM/SIAM*, 2002, pages 657–666.
- [NN12] Gonzalo Navarro and Yakov Nekrich. “Top- $k$  document retrieval in optimal time and linear space”. In: *SODA. SIAM*, 2012, pages 1066–1077. DOI: [10.1137/1.9781611973099.84](https://doi.org/10.1137/1.9781611973099.84).
- [Wei73] Peter Weiner. “Linear Pattern Matching Algorithms”. In: *SWAT (FOCS)*. IEEE Computer Society, 1973, pages 1–11. DOI: [10.1109/SWAT.1973.13](https://doi.org/10.1109/SWAT.1973.13).