A Scalable Index for Top-k Subtree Similarity Queries

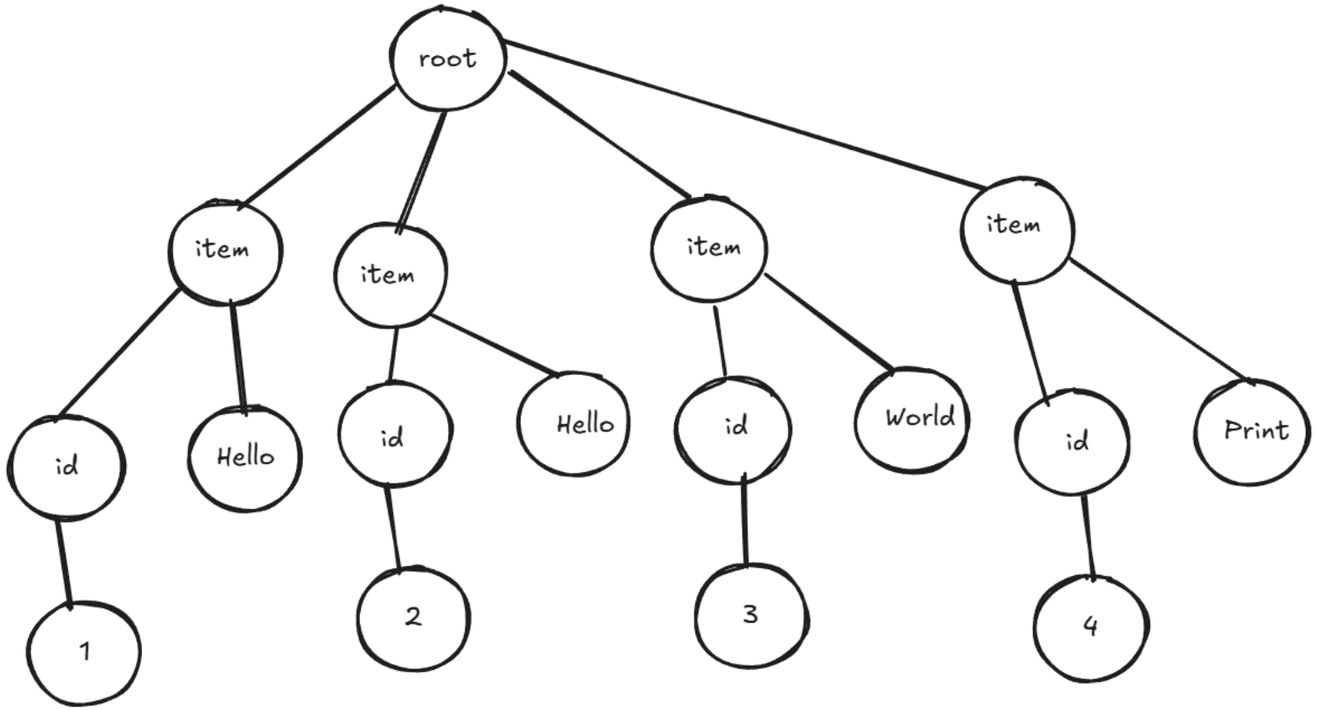
Presented by: Aliza Subedi, Bishal Bashyal

FSU | FLORIDA STATE UNIVERSITY

Problem Statement

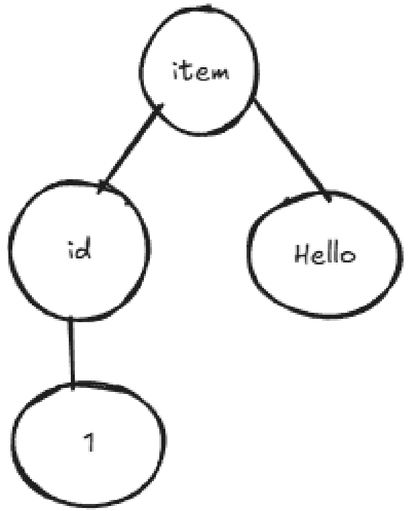
```
<root>
  <item id="1">Hello</item>
  <item id="1">Hello</item>
  <item id="2">World</item>
  <item id="3">World</item>
  <item id="4">Print</item>
</root>
```

Document T



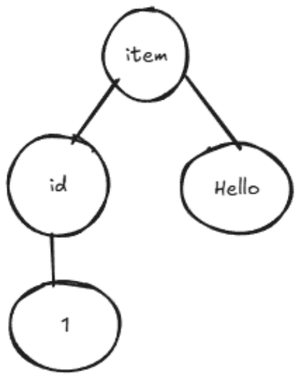

```
<item id="1">Hello</item>
```

Query Q

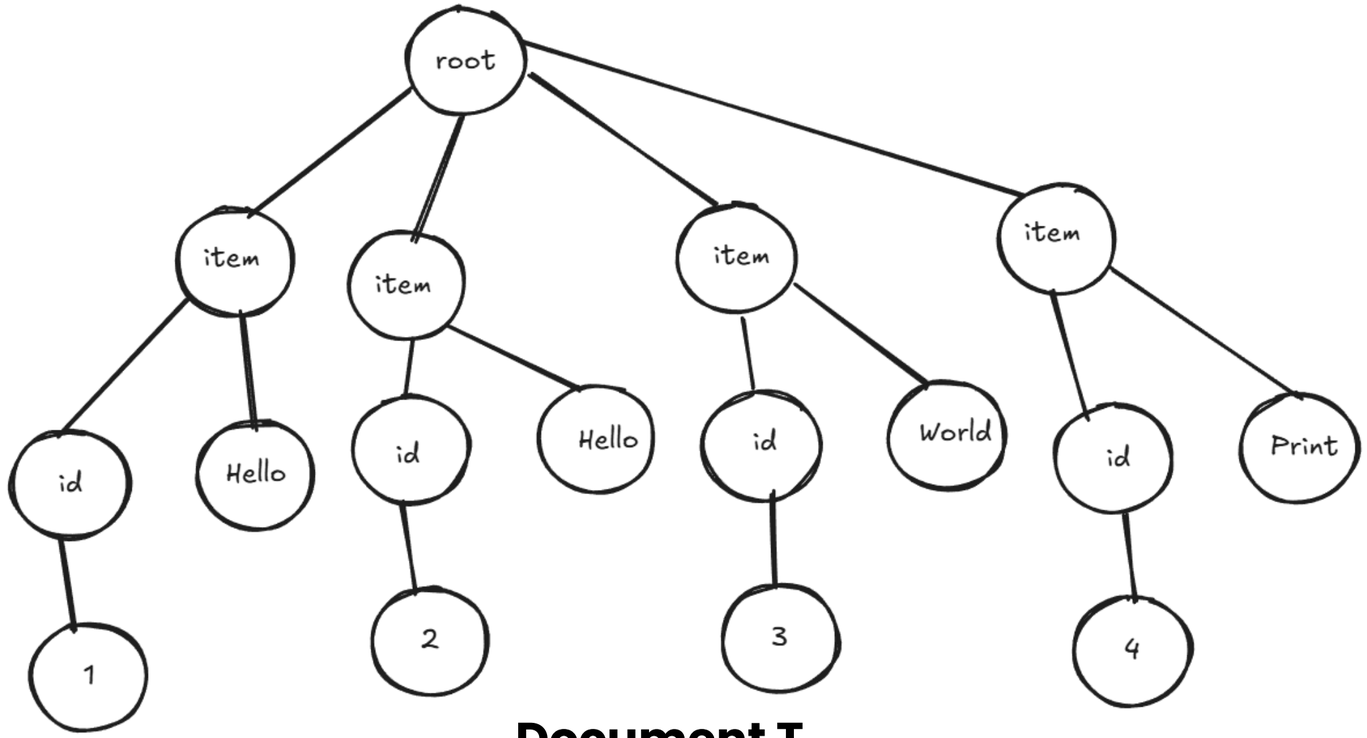


Problem Statement

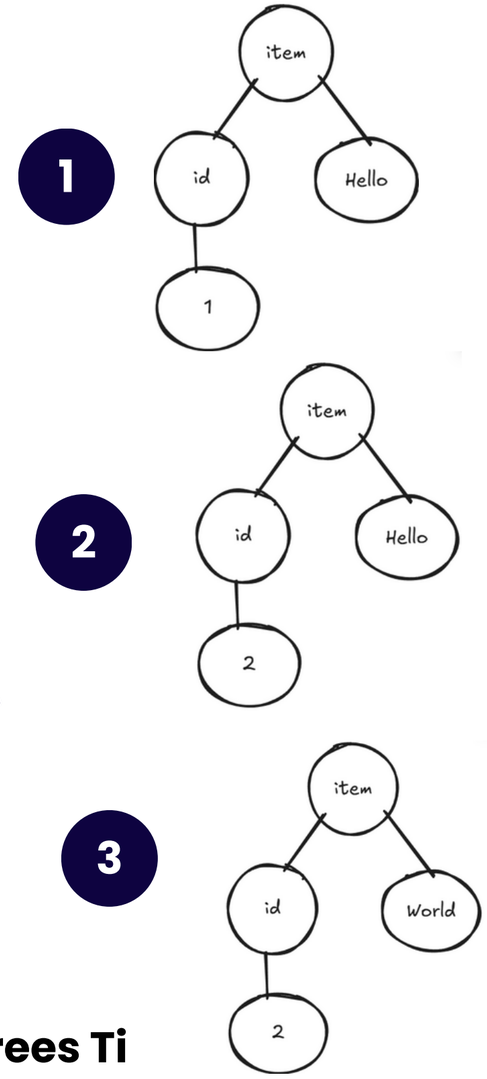
Given a small tree Q and a number K, Find the K subtrees from the document tree T which is most similar to Q.



Query Q



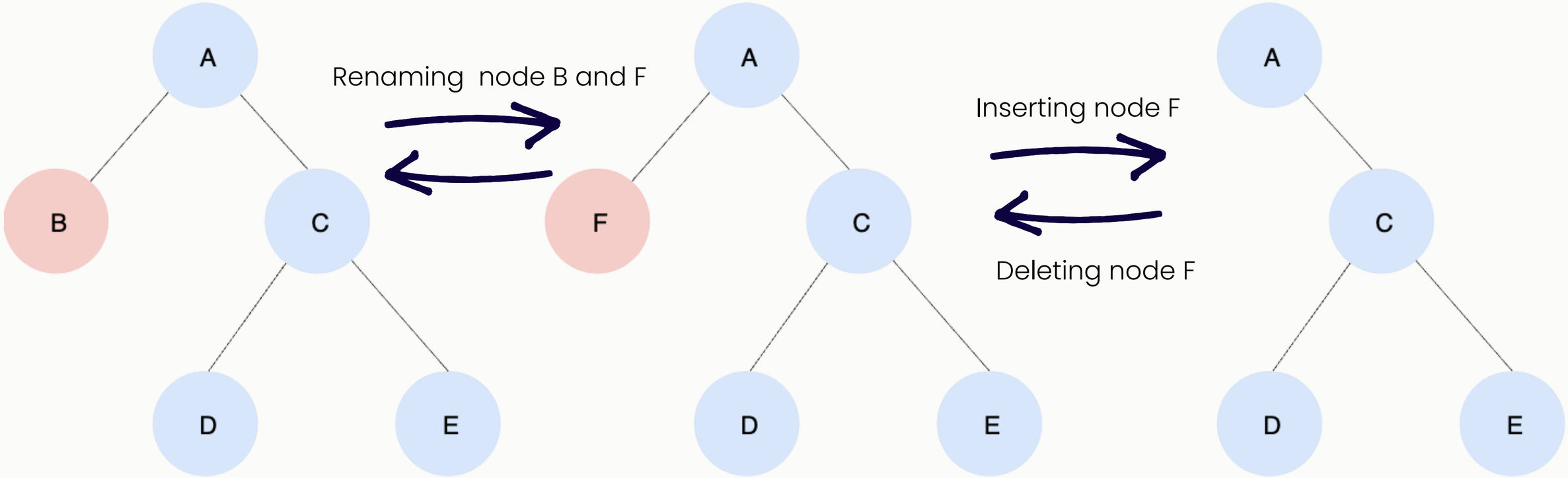
Document T



Subtrees Ti

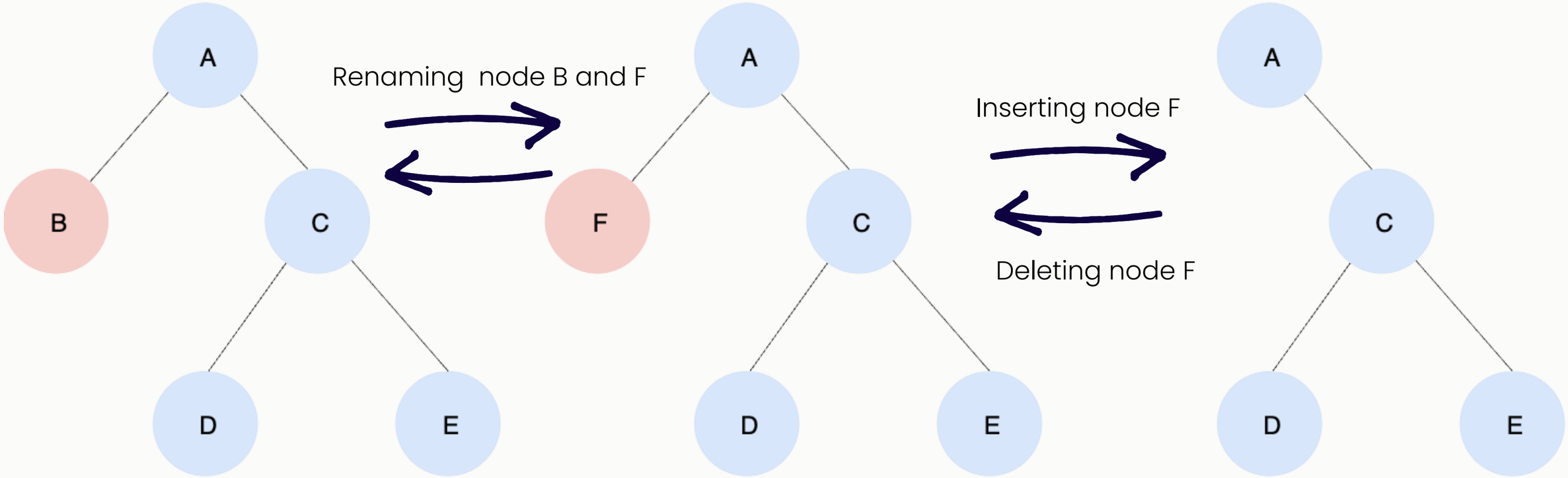
Tree Edit Distance

Minimum number of node edit operations that transform one tree to another.



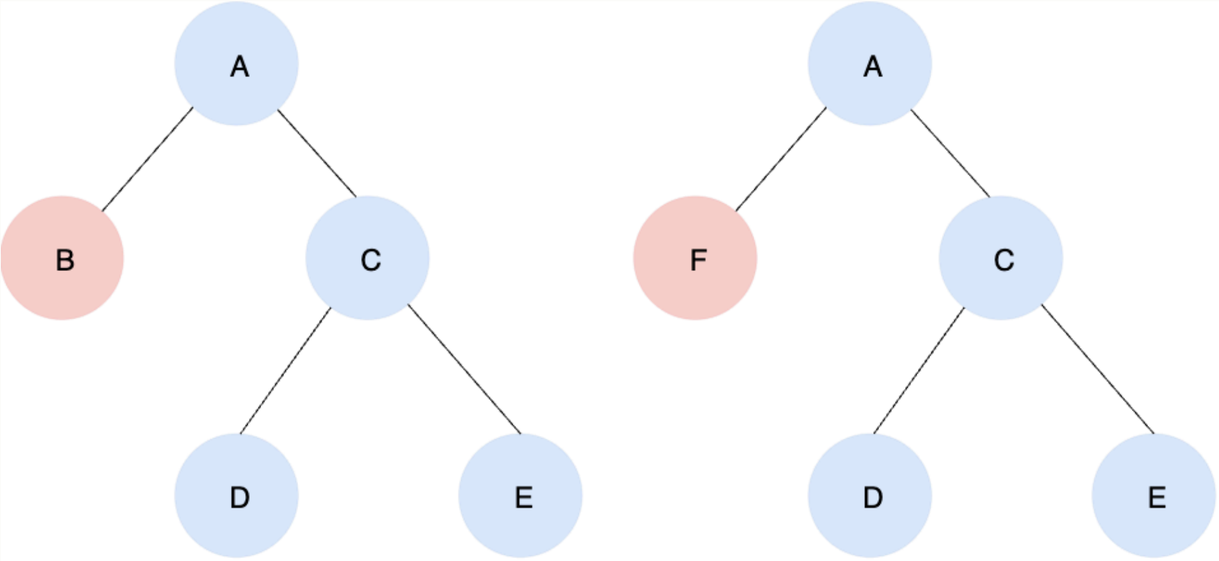
Tree Edit Distance

Minimum number of node edit operations that transform one tree to another.



Time Complexity: O(n^3) Space Complexity: O(n^2)

Cost Estimate and Bounds

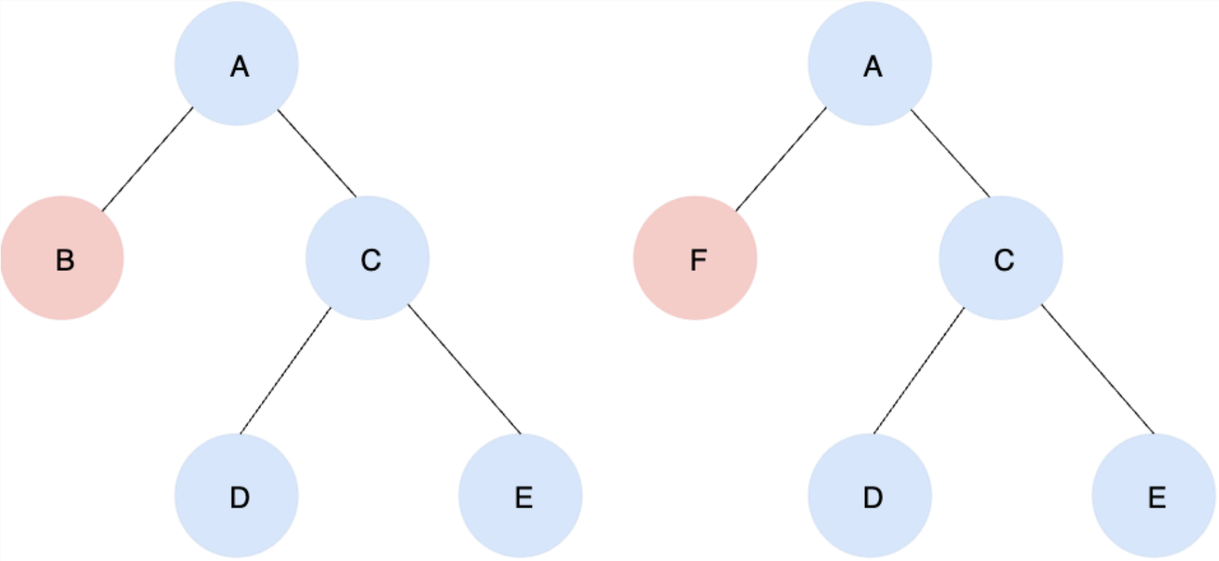


Document Subtree (T)

Query (Q)

- **Label Lower Bound (llb):** how many labels are not similar in both trees ?
- **Size Lower Bound (slb):** Absolute size difference between T and Q.

Cost Estimate and Bounds



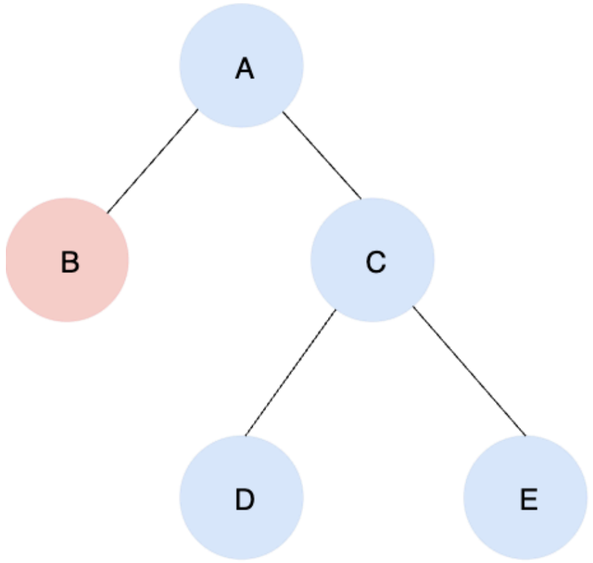
Document Subtree (T)

Query (Q)

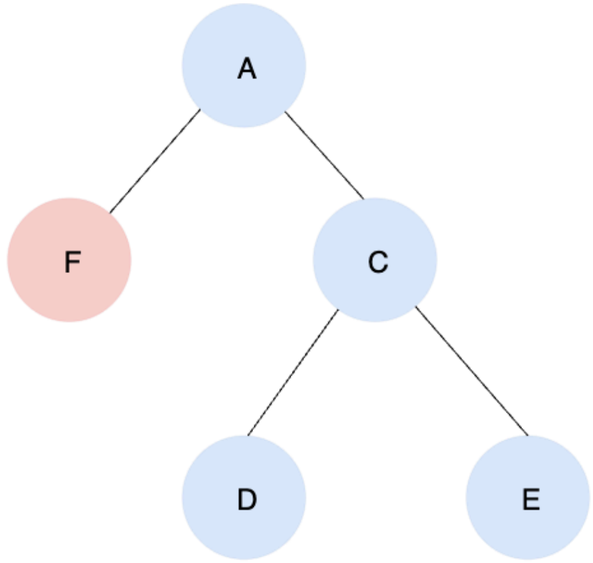
- **Label Lower Bound (llb):** how many labels are not similar in both trees ?
- **Size Lower Bound (slb):** Absolute size difference between T and Q.

- Query labels = {A, C, D, E, F}
- Tree labels = {A, C, D, E, B}
- LLB = 1 (since F is missing)

Cost Estimate and Bounds



Document Subtree (T)



Query (Q)

- **Label Lower Bound (llb):** how many labels are not similar in both trees ?
- **Size Lower Bound (slb):** Absolute size difference between T and Q.

- Query labels = {A, C, D, E, F}
- Tree labels = {A, C, D, E, B}
- LLB = 1 (since F is missing)

- Number of Query nodes = 5
- Number of Tree nodes = 5
- SLB = 0 (size of both trees are same)

State of the Art Methods

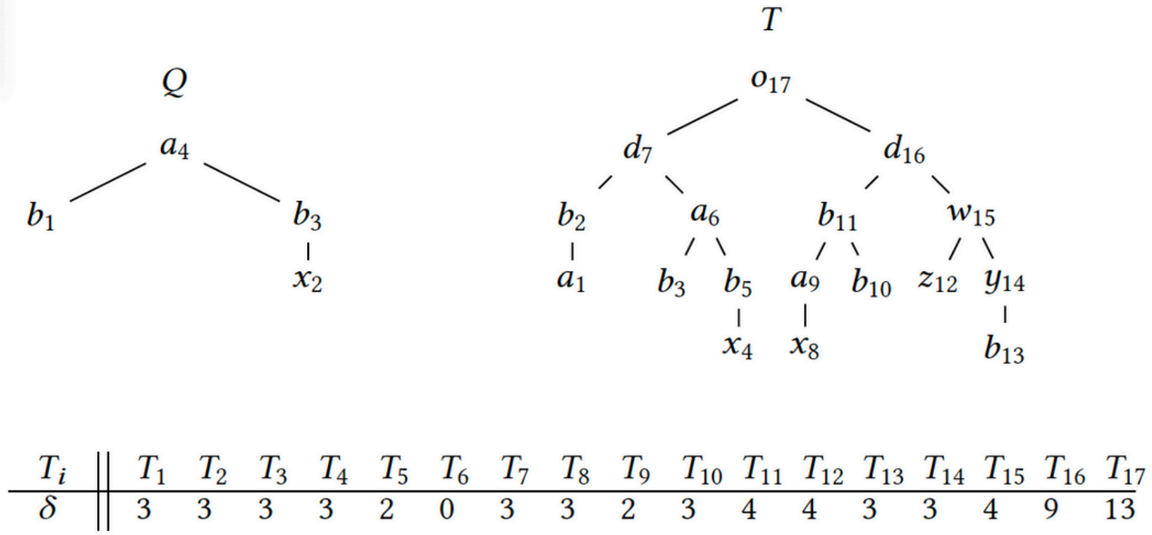
	Tasm-Postorder	**Structure Search**
Index	Index-Free method	Index-based method
Querying	Slow	Fast
Memory Usage	Low	HIgh (quadratic)
Index Updates	-	No

Bounding Condition

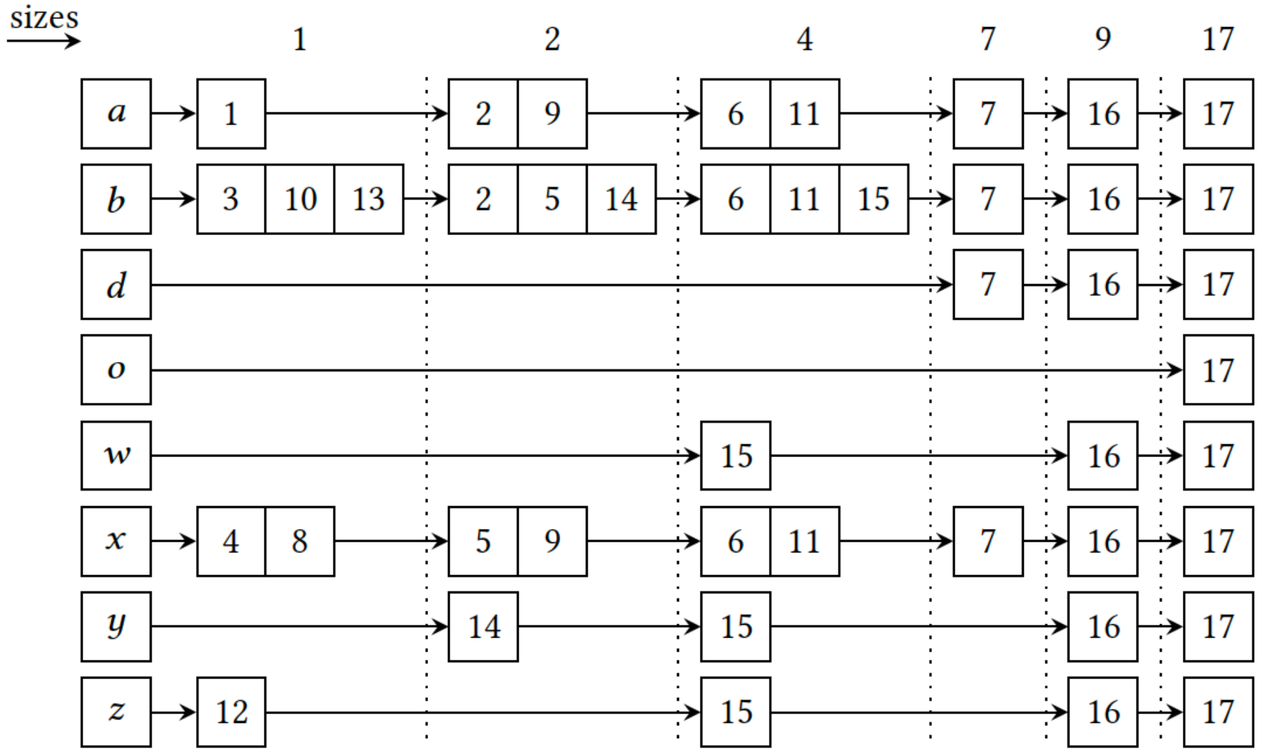
$$\textbf{LowerBound}(Q, T_i) \geq \delta(R'[k])$$

Avoids unnecessary Tree Edit Distance calculations by eliminating subtrees early if they cannot match the query within the allowed edit distance.

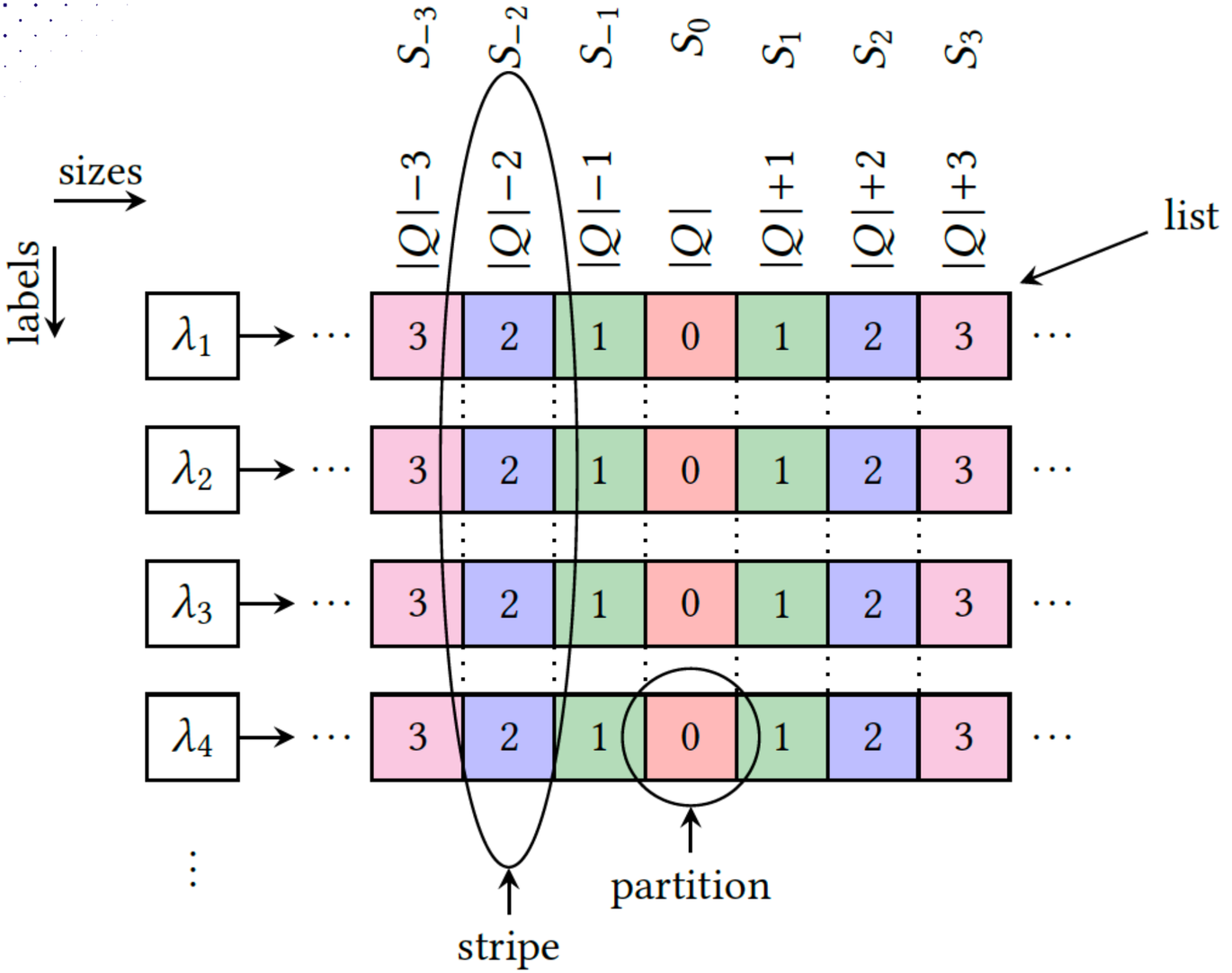
Inverted Index



(a) Example of Query Tree (Q) and Document Tree (T)

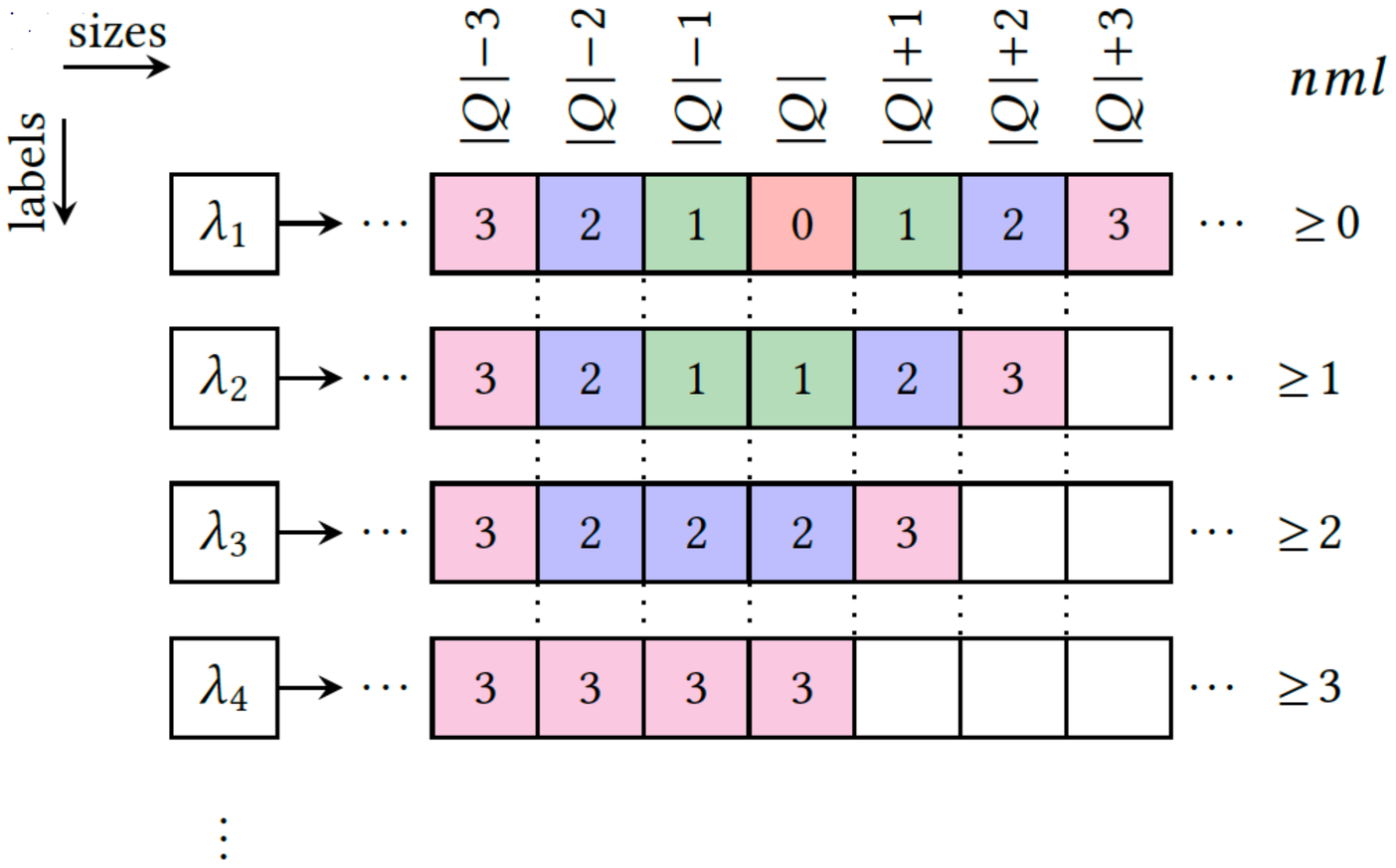


(b) Inverted List index of T



MergeAll ALgorithm

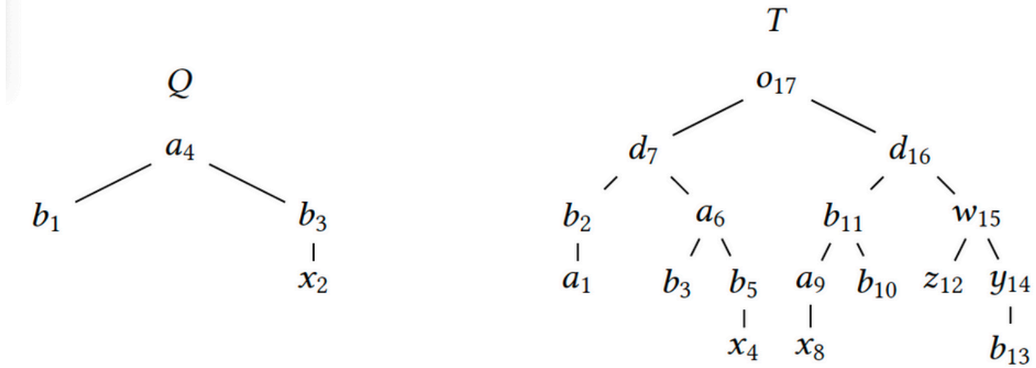
- Start from the S0 stripe whith SLB = 0 (possible top ranking).
- Compute llb(Q,Ti) aand if llb(Q,Ti) != slb(Q,Ti), cache that Ti to a bucket b[j] where j is the stripe number.
- If llb = slb, compute TED and update the ranking list.
- When evaluating a new stripe, evaluate bucket b[j] first.
- As processing continues, check if |R| = k and j >= δ (R [k]) for terminating the filtering process.



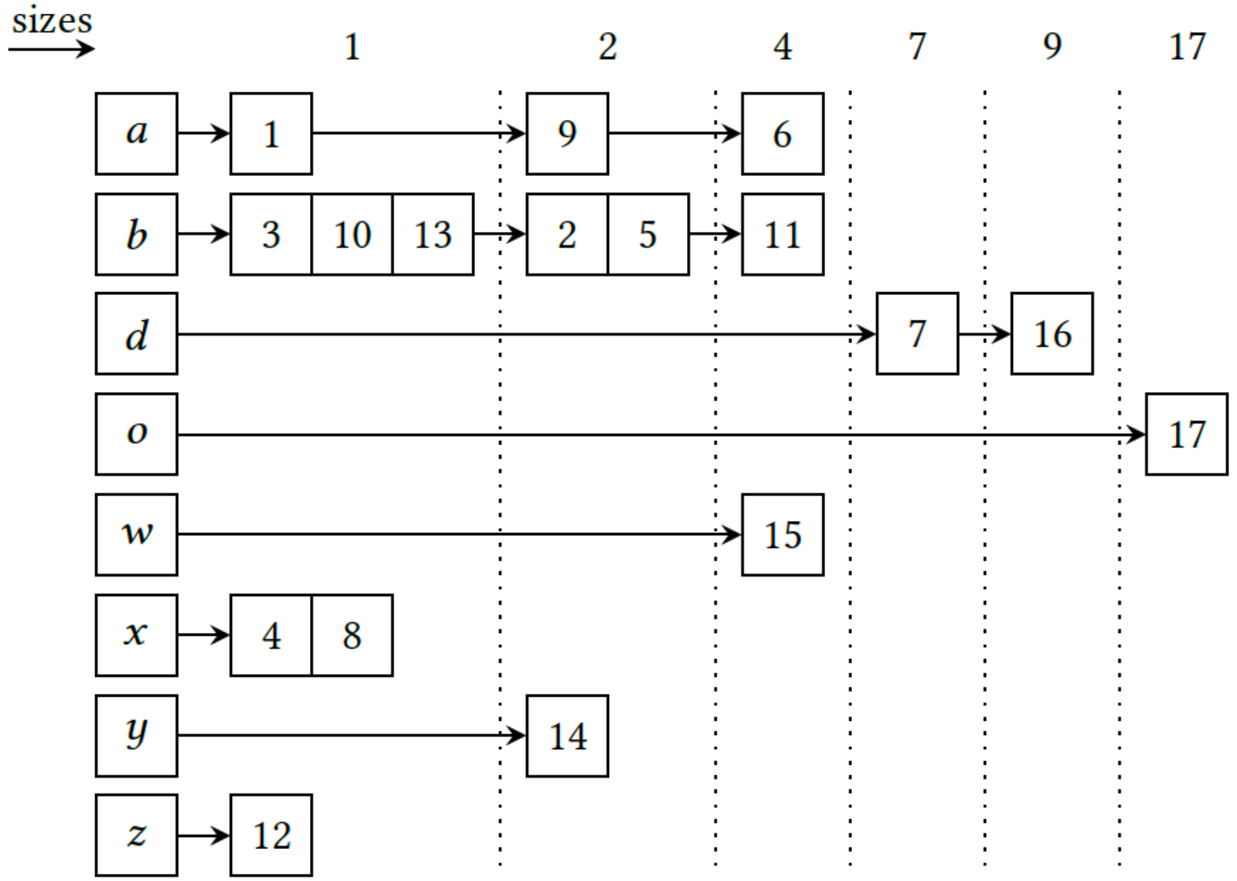
Cone Filtering

- Mergeall could be inefficient by processing large partitions within a stripe.
- Cone Accesses short, sorted lists first, often terminating before reaching large partitions. Processes only a subset of partitions unlike mergeall.
- Uses min = |Q| - B , max = |Q| + B - nml (Ti) values in each round.
- Terminates if |R| = k and B >= δ (R [k]).
- Quadratic Space Complexity.

Slim Inverted list

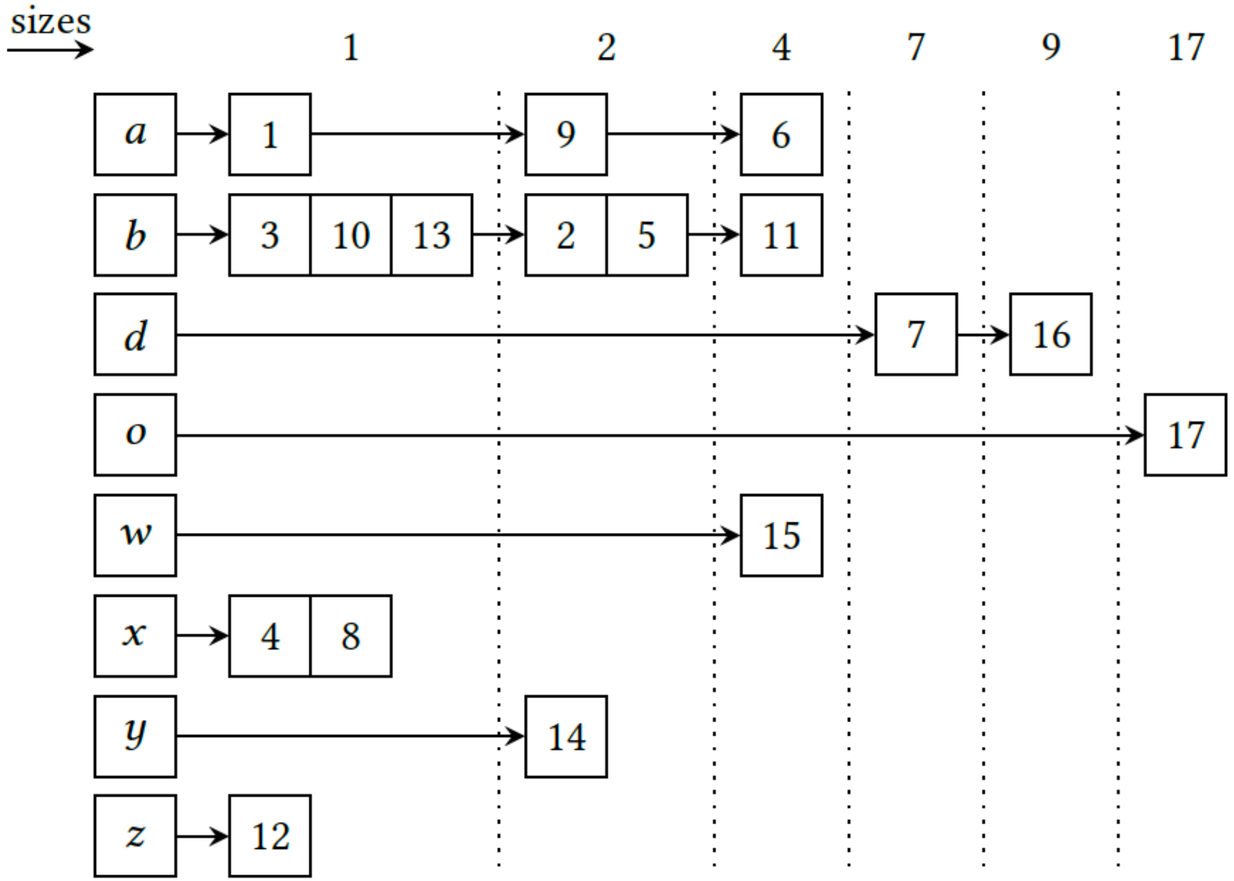


T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}
δ	3	3	3	3	2	0	3	3	2	3	4	4	3	3	4	9	13



(a) Slim inverted list index.

SlimCone

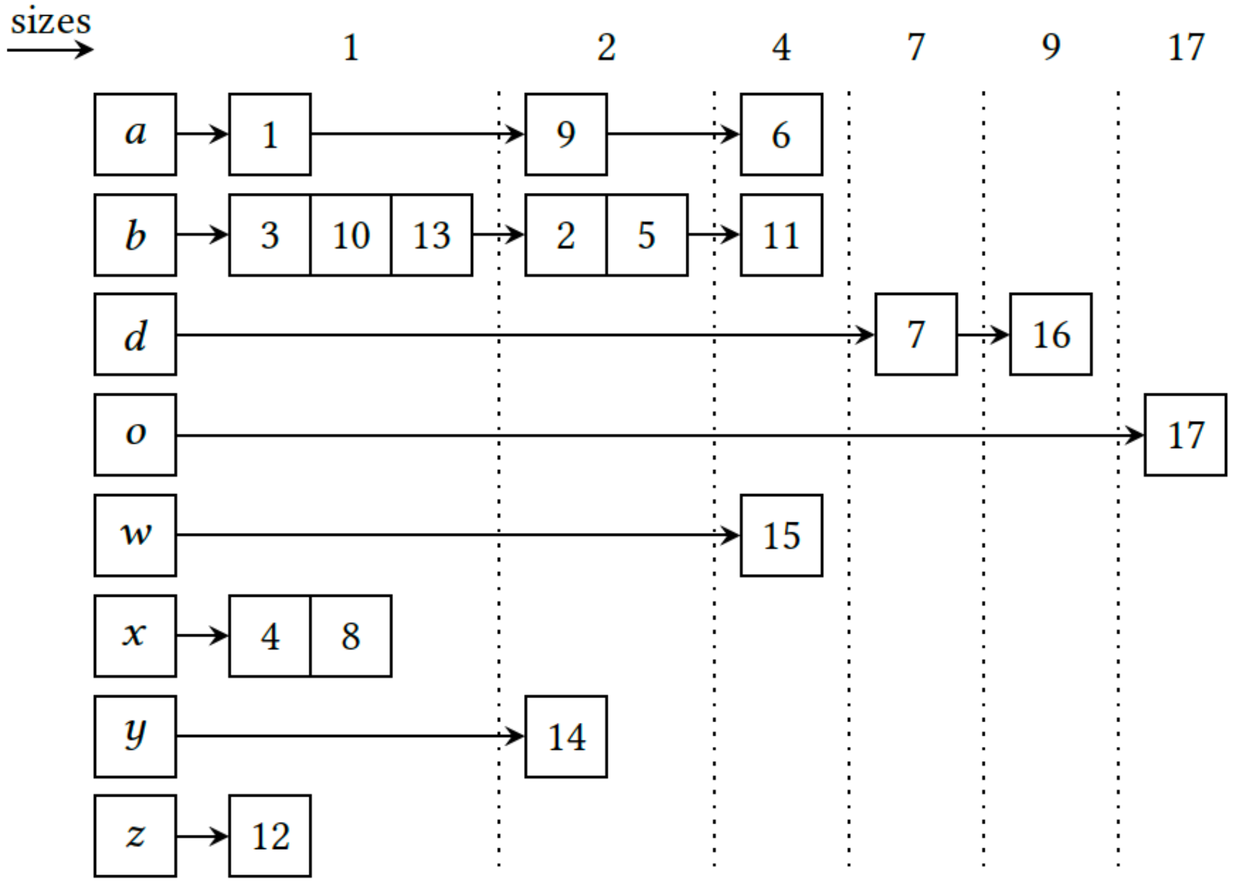


(a) Slim inverted list index.

Algorithm

- Reconstruct subtrees on the fly by unveiling.
- Round Based with B=0, and increases in each round.
- Generate possible candidates by calculating llb(Q,Ti).
- If(llb(Q,Ti) = B), evauate TED, else cache.
- Check if $|R| = k \wedge B \geq \delta(R[k]$ for terminating filtering algorithm.
- Works with a Linear space (slim) index.
- Implements sliminvertedlist as a Binary Search tree to support index updates.

Supporting Updates



(a) Slim inverted list index.

Dynamic Node Index

Each node entry contains:

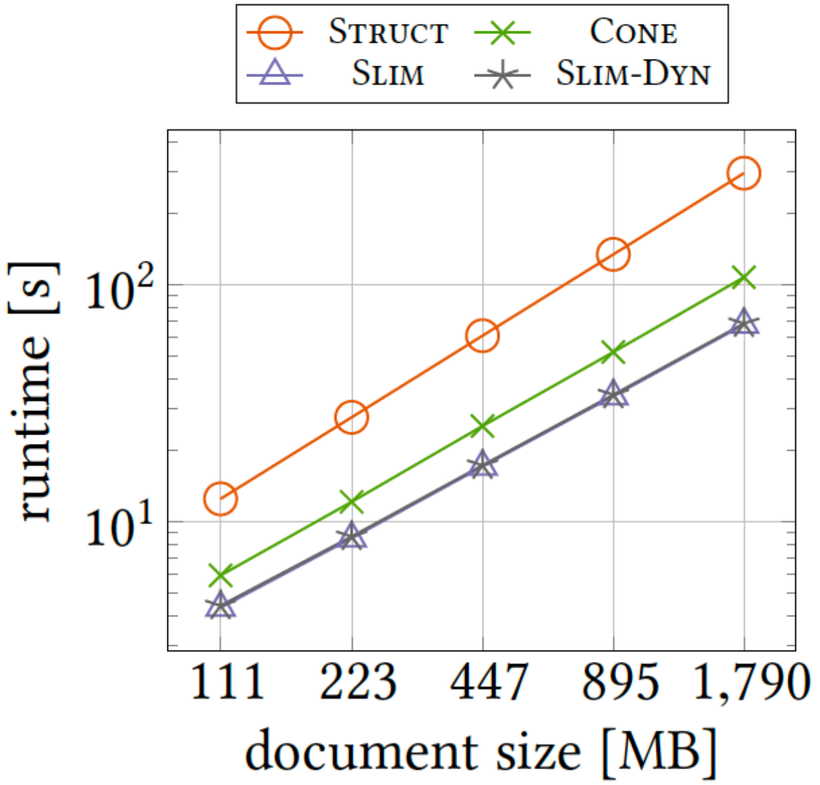
- lbl: The node's label (e.g., a,b,x a, b, x a,b,x).
- size: The size of the subtree rooted at the node.
- par: The parent node's identifier.
- c1: The identifier of the first child.
- sib: The identifier of the next (right) sibling.
- No need to reconstruct the entire index.

Comparision

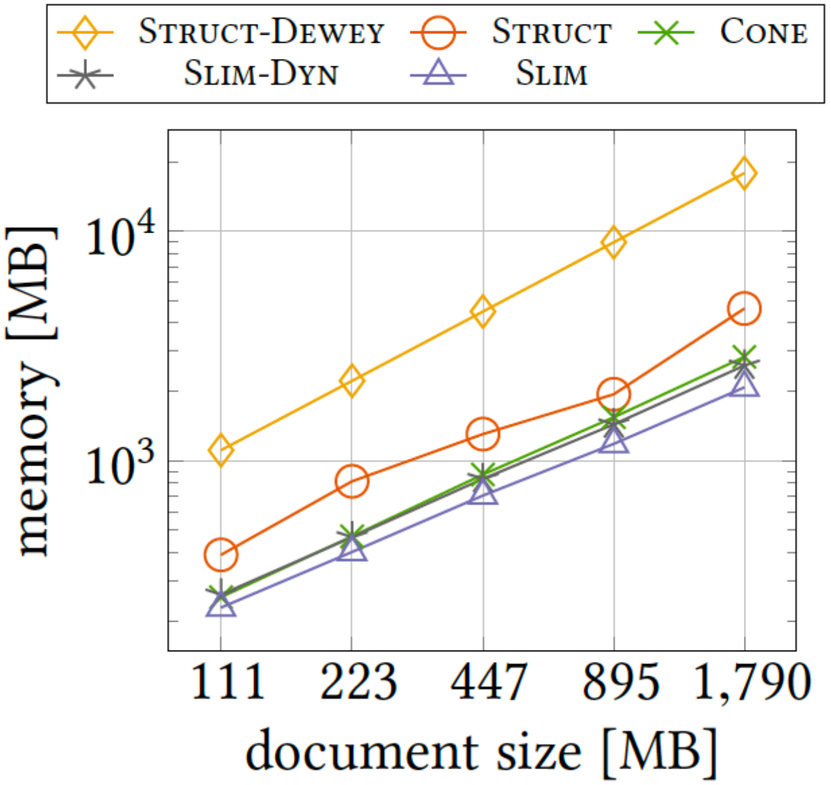
	MERGEALL	CONE	SLIMCONE	SLIMCONE - DYN
Handles large datasets efficiently	✗	✓	✓	✓
Space-efficient index (linear size)	✗	✗	✓	✓
Faster Index Build	✗	✗	✓	✓
Supports dynamic updates	✗	✗	✗	✓

Datasets

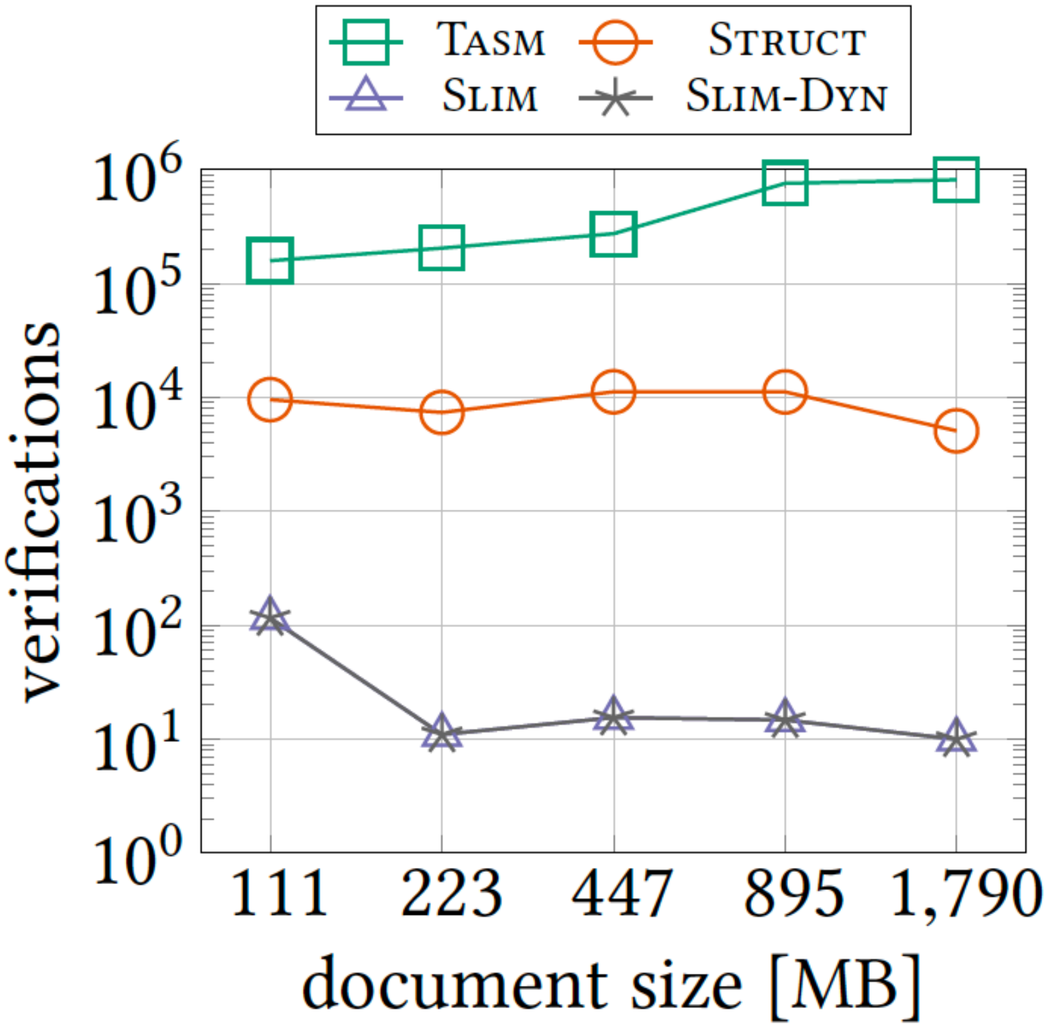
Name	Size T [MB]	Size [Nodes]	
		$\|T\|$	avg. $\|T_i\|$
XMark1	112	$3.6 \cdot 10^6$	6.2
XMark2	223	$7.2 \cdot 10^6$	6.2
XMark4	447	$14.4 \cdot 10^6$	6.2
XMark8	895	$28.9 \cdot 10^6$	6.2
XMark16	1,790	$57.8 \cdot 10^6$	6.2
TreeBank	83	$3.8 \cdot 10^6$	8.4
DBLP	2,161	$126.5 \cdot 10^6$	3.4
SwissProt	6,137	$479.3 \cdot 10^6$	5.1



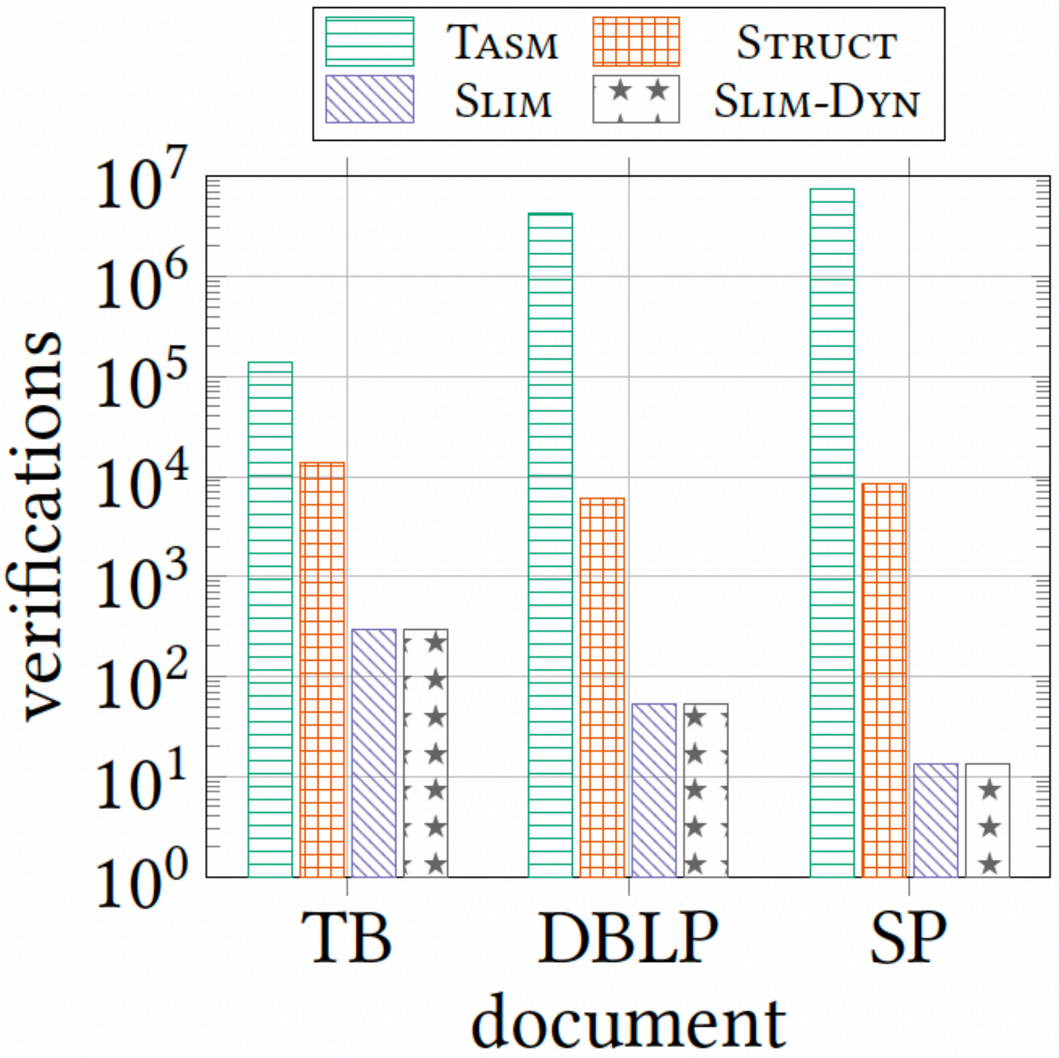
(a) Build time, XMark.



(b) Index size, XMark.



Xmark dataset, |Q| =16, K=10



|Q| =16, K=10

Slimcone

Efficient

Scalable

Linear Space indexing

Updatable

Conclusion

- Proposed a scalable and novel solution for top-k subtree similarity query retrieval.
- Presented baseline algorithm, Mergeall and cone and builds slim on top of it.
- Introduced slim index reducing memory complexity from quadratic to linear.
- Extended Slimcone to Slim-Dyn for dynamic document updates.
- Validated performance through extensive evaluation on synthetic and real-world datasets.

Thank you!