

Information Retrieval

Venkatesh Vinayakarao
venkateshv@cmi.ac.in

Chennai Mathematical Institute



What we find changes who we become.

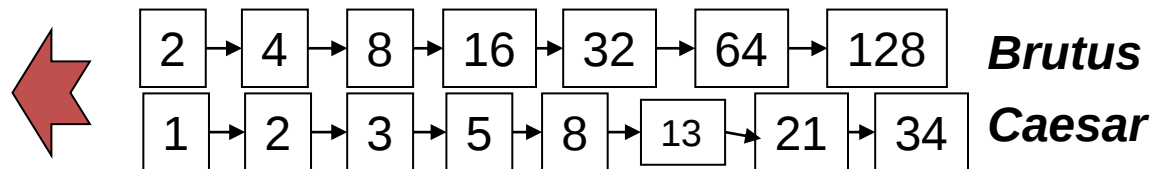
-Peter Morville.



Query Processing with Inverted Index

Query processing: AND

- Consider processing the query:
Brutus AND Caesar
 - Locate ***Brutus*** in the Dictionary;
 - Retrieve its postings.
 - Locate ***Caesar*** in the Dictionary;
 - Retrieve its postings.
 - “Merge” the two postings (intersect the document sets):



Common Interview Question

- <https://www.geeksforgeeks.org/intersection-of-two-sorted-linked-lists/>

GeeksforGeeks
A computer science portal for geeks

[G](#)[Algo ▼](#)[DS ▼](#)[Languages ▼](#)[Interview ▼](#)[Students ▼](#)[GATE ▼](#)[CS Subjects ▼](#)[Quizzes ▼](#)

Geeks Classes

Quick Links for Sorting

[Sorting Terminology](#)[Stability in sorting algorithms](#)[Time Complexities of all Sorting Algorithms](#)[External Sorting](#)

Intersection of two Sorted Linked Lists

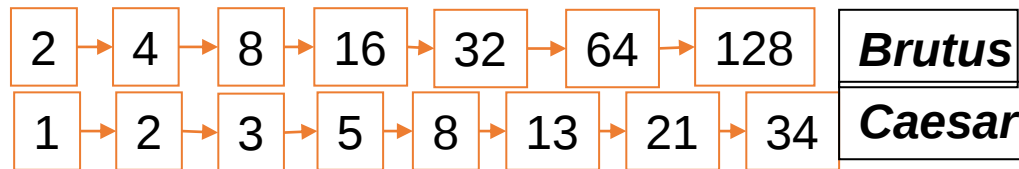


Given two lists sorted in increasing order, create and return a new list representing the intersection of the two lists. The new list should be made with its own memory — the original lists should not be changed.

For example, let the first linked list be 1->2->3->4->6 and second linked list be 2->4->6->8, then your function should create and return a third list as 2->4->6.

The Merge

- Walk through the two postings simultaneously. Use two pointers.

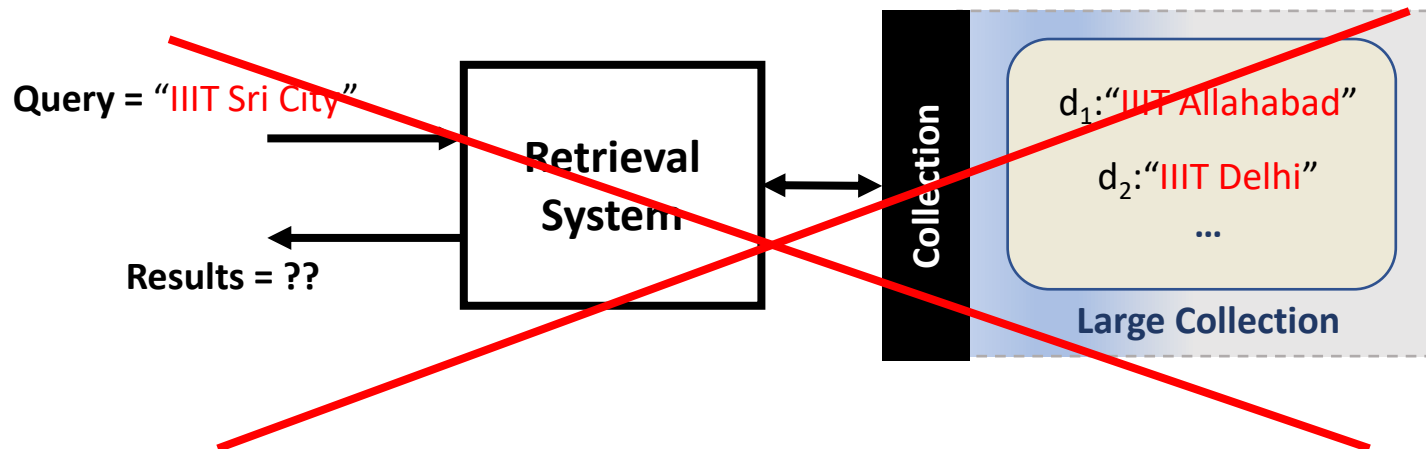


Note that the postings need to be sorted by document ID.

- If the list lengths are m and n , the merge takes $O(m + n)$ operations.

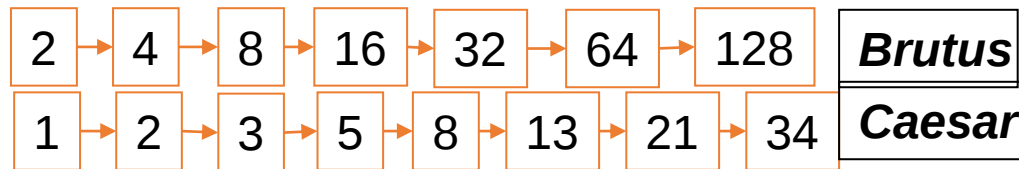
The Big Picture

- Content Processing
 - ~~Build Term Document Matrix~~
 - Build Inverted Index
- Query Handling
 - ~~Boolean AND~~
 - Intersect the Posting Lists (called merging process)



The Merge

- Walk through the two postings simultaneously. Use two pointers.



Note that the postings need to be sorted by document ID.

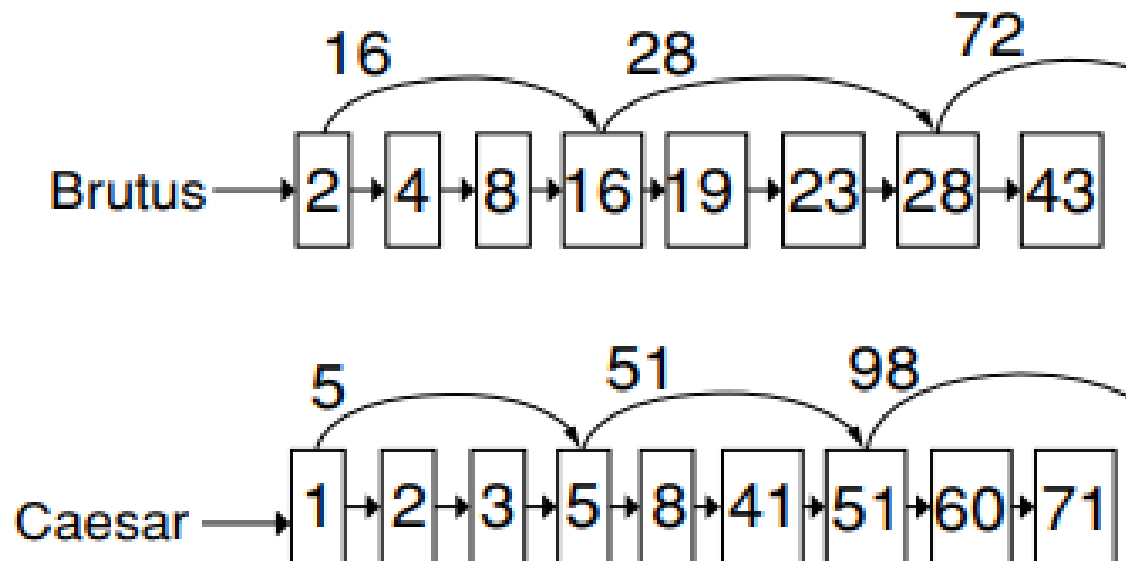
- If the list lengths are x and y , the merge takes $O(x+y)$ operations.

Can we do better?

Inspired from multiple index idea of DBMS

Skip List

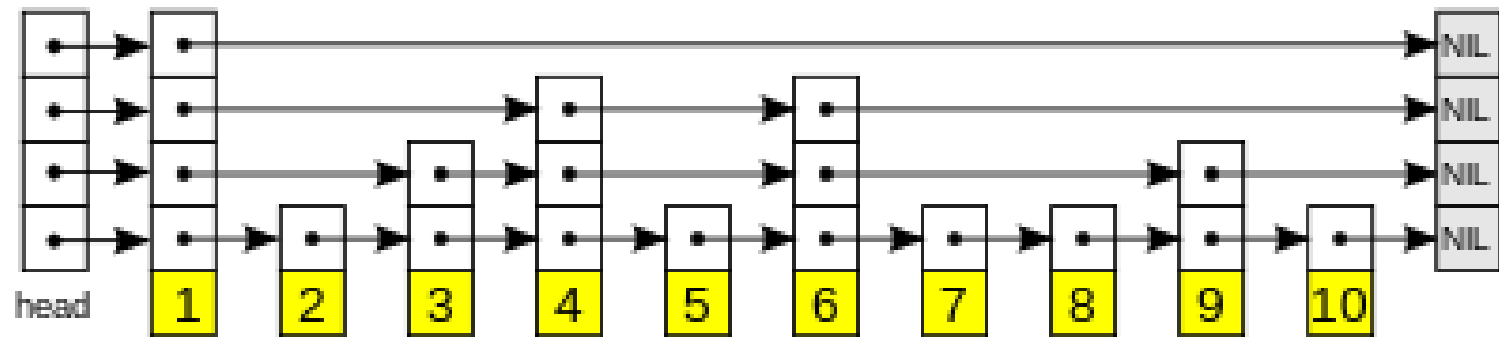
- *Skip lists* are postings lists with skip pointers



Skip Lists

- Advantages
 - Achieves sublinear list intersection time which is better than $O(m+n)$.
 - Works better for static index. Updates are expensive.
 - Useful for “AND” query.
- Disadvantages
 - Needs space to store skip pointers.
 - The effectiveness depends on where we place the skips.
 - More skips → Shorter skip interval (and more comparisons).
 - Less skips → Less extra space consumed but more linear traversal (lesser comparison). We have a tradeoff.

A Variant of Skip List

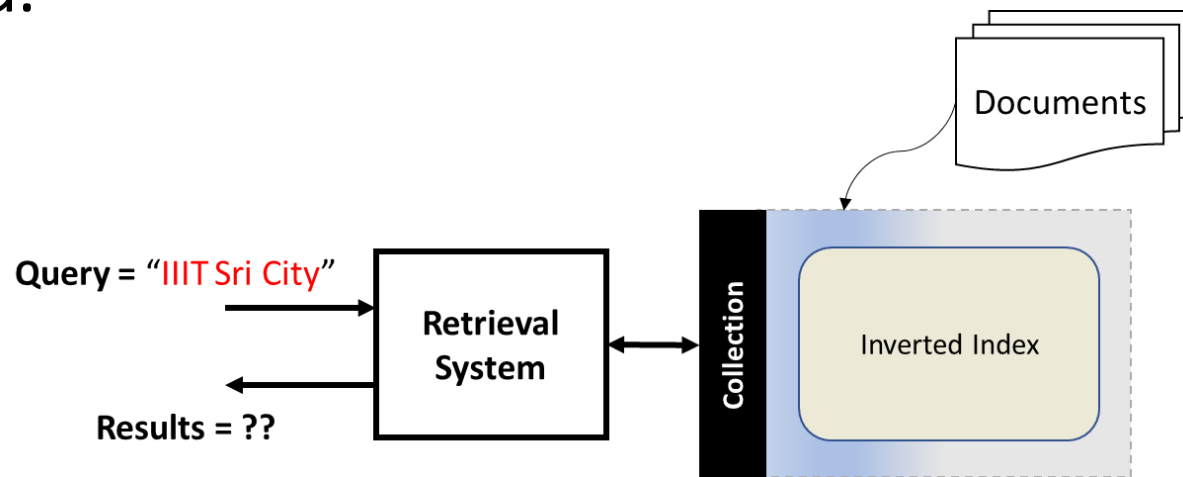


Multiple levels of skips

Lucene implements a similar multi-level skip list

Answering Phrasal Queries

- Sometimes, we are looking for a phrase and not a word.



- Here, what if we do not want to match "IIIT Delhi" even if "IIIT" exists in "IIIT Delhi"?

One (bad) Approach

- Index all biwords
 - Friends, Romans, Countrymen → Friends Romans, Romans Countrymen
- How do you match the query IIIT Sri City, Chittoor?
 - **“IIIT Sri”** AND **“Sri City”** AND **“City Chittoor”** must exist.
- The Problem: **“IIIT”** AND **“Sri City”** AND **“Chittoor”** sounds like a much better query!
 - Natural Language Processing techniques can help in query formulation.

A Better Approach

- Store Positional Information

<term, number of docs containing term;

doc1: position1, position2 ... ;

doc2: position1, position2 ... ;

etc.>

Extended Boolean Model with Positional Index

← “to” appears 993K times overall.

to, 993427:

$\langle 1, 6: \langle 7, 18, 33, 72, 86, 231 \rangle;$ ← “to” appears six times in doc 1
2, 5: $\langle 1, 17, 74, 222, 255 \rangle;$ at positions 7, 18,
4, 5: $\langle 8, 16, 190, 429, 433 \rangle;$
5, 2: $\langle 363, 367 \rangle;$
7, 3: $\langle 13, 23, 191 \rangle; \dots \rangle$

be, 178239:

$\langle 1, 2: \langle 17, 25 \rangle;$
4, 5: $\langle 17, 191, 291, 430, 434 \rangle;$
5, 3: $\langle 14, 19, 101 \rangle; \dots \rangle$

Which document is likely to contain “to be”?

Doc 4 at positions (190, 191), (429,430) (433, 434)

Proximity Search

- IIIT /3 Chittoor
 - /k means “within k words of”
- Merging postings is expensive
 - Index well-known phrases such as “Taj Mahal”

Combination Schemes

- biword index and positional index ideas can be combined.
- Use biword index or common phrases (such as Taj Mahal).
 - Avoids merging postings lists.
- Use positional index for other phrases (such as IIIT Chittoor).

Index Types

Forward Index

doc # 12 0 1 2 3 4

Mary	has	a	little	lamb
------	-----	---	--------	------

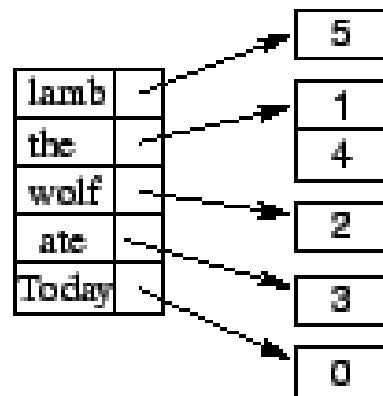
doc # 13
 (old) 0 1 2 3 4

The	wolf	ate	the	lamb
-----	------	-----	-----	------

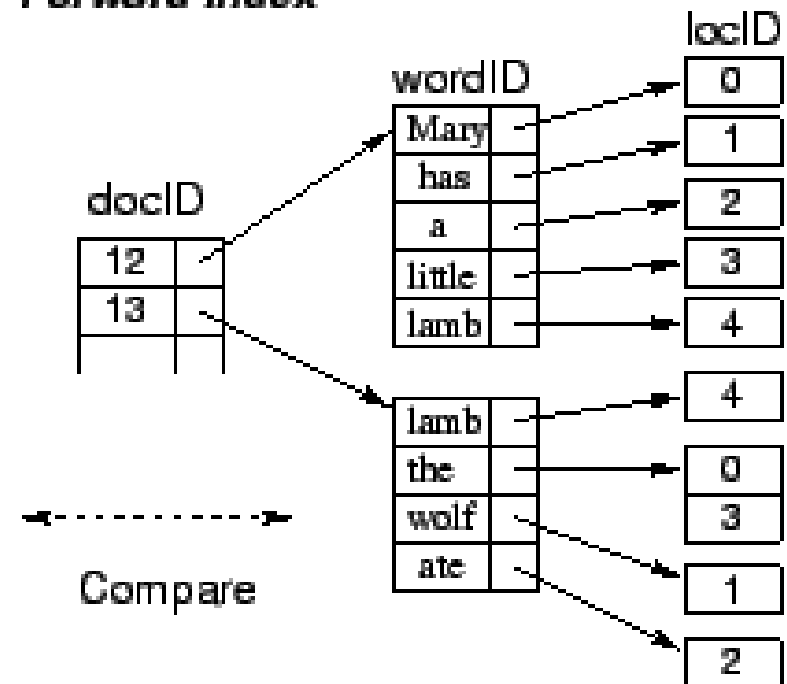
doc # 13
 (new) 0 1 2 3 4 5

Today	the	wolf	ate	the	lamb
-------	-----	------	-----	-----	------

Construct
Forward index



Forward Index



Inverted Index

Doc 1

I did enact Julius Caesar: I was killed
i' the Capitol; Brutus killed me.

term	docID	term	docID
I	1	ambitious	2
did	1	be	2
enact	1	brutus	1
julius	1	brutus	2
caesar	1	capitol	1
I	1	caesar	1
was	1	caesar	2
killed	1	caesar	2
i'	1	did	1
the	1	enact	1
capitol	1	hath	1

Doc 2

So let it be with Caesar. The noble I
hath told you Caesar was ambitious:

Dictionary

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2

Inverted Positional Index

- Query: “ $to_1 be_2 or_3 not_4 to_5 be_6$ ”
 - TO, 993427:
 - $\langle 1: \langle 7, 18, 33, 72, 86, 231 \rangle;$
 - $2: \langle 1, 17, 74, 222, 255 \rangle;$
 - $4: \langle 8, 16, 190, 429, 433 \rangle;$
 - $5: \langle 363, 367 \rangle;$
 - $7: \langle 13, 23, 191 \rangle; \dots \rangle$
 - BE, 178239:
 - $\langle 1: \langle 17, 25 \rangle;$
 - $4: \langle 17, 191, 291, 430, 434 \rangle;$
 - $5: \langle 14, 19, 101 \rangle; \dots \rangle$
- Document 4 is a match!

Inverted Positional Index

1	The old night keeper keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

Table with 6 documents

<< index >>

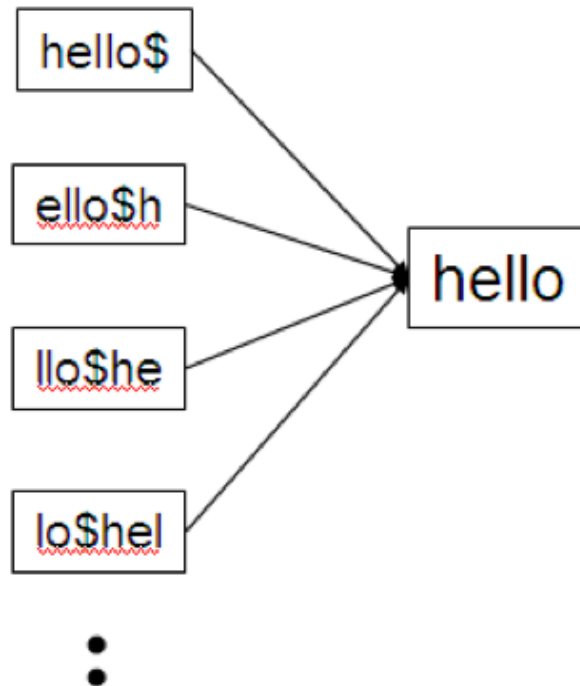
?

Lucene

Term	Freq	Postings & Positions
and	1	(6,1) (6,6)
big	2	(2,3) (2,8) (3,8)
dark	1	(6,5)
did	1	(4,7)
gown	1	(2,10)
had	1	(3,6)
house	2	(2,5) (3,2)
in	5	<(1,8)> <(2,1)> <(2,6)> <(3,3)> <(5,7)> <(6,3)> <(6, 8)>
keep	3	(1,7) (3,10) (5,6)
keeper	3	(1,4) (4,5) (5,3)
keeps	3	(1,5) (5,4) (6,2)
light	1	(6,10)
never	1	(4,6)
night	3	(1,3) (4,4) (5,2) (5,9)
old	4	(1,2) (2,4) (2,9) (3,9) (4,3)
sleep	1	(4,8)
sleeps	1	(6,7)
the	6	<(1,1)> <(1,6)> <(2,2)> <(2,7)> <(3,1)> <(3,4)> <(3,7)> <(4,2)> <(5,1)> <(5,5)> <(5,8)> <(6,4)> <(6,9)>
town	2	(1,10) (3,5)
where	1	(4,1)

Permuterm index

- For HELLO, we've stored: *hello\$, ello\$h, llo\$he, lo\$hel, and o\$hell*



K-gram Index

- Enumerate all character k -grams (sequence of k characters) occurring in a term
 - Example: from “April is the cruelest month” we get the bigrams: \$a
ap pr ri il l\$ \$i is s\$ \$t th he e\$ \$c cr ru ue el le es st t\$ \$m mo on nt
h\$
- \$ is a special word boundary symbol, as before.

A partial snapshot of a sample 3-gram index



Quiz

- How does the bi-gram inverted non-positional index look like?
 - Assume:
 - Collection: d1: INDIA, d2: ASIA

Thank You