Monsoon 2020

3 - Boolean Retrieval

Information Retrieval

by

Dr. Rajendra Prasath

Indian Institute of Information Technology
Sri City – 517 646, Andhra Pradesh, India

Boolean Retrieval

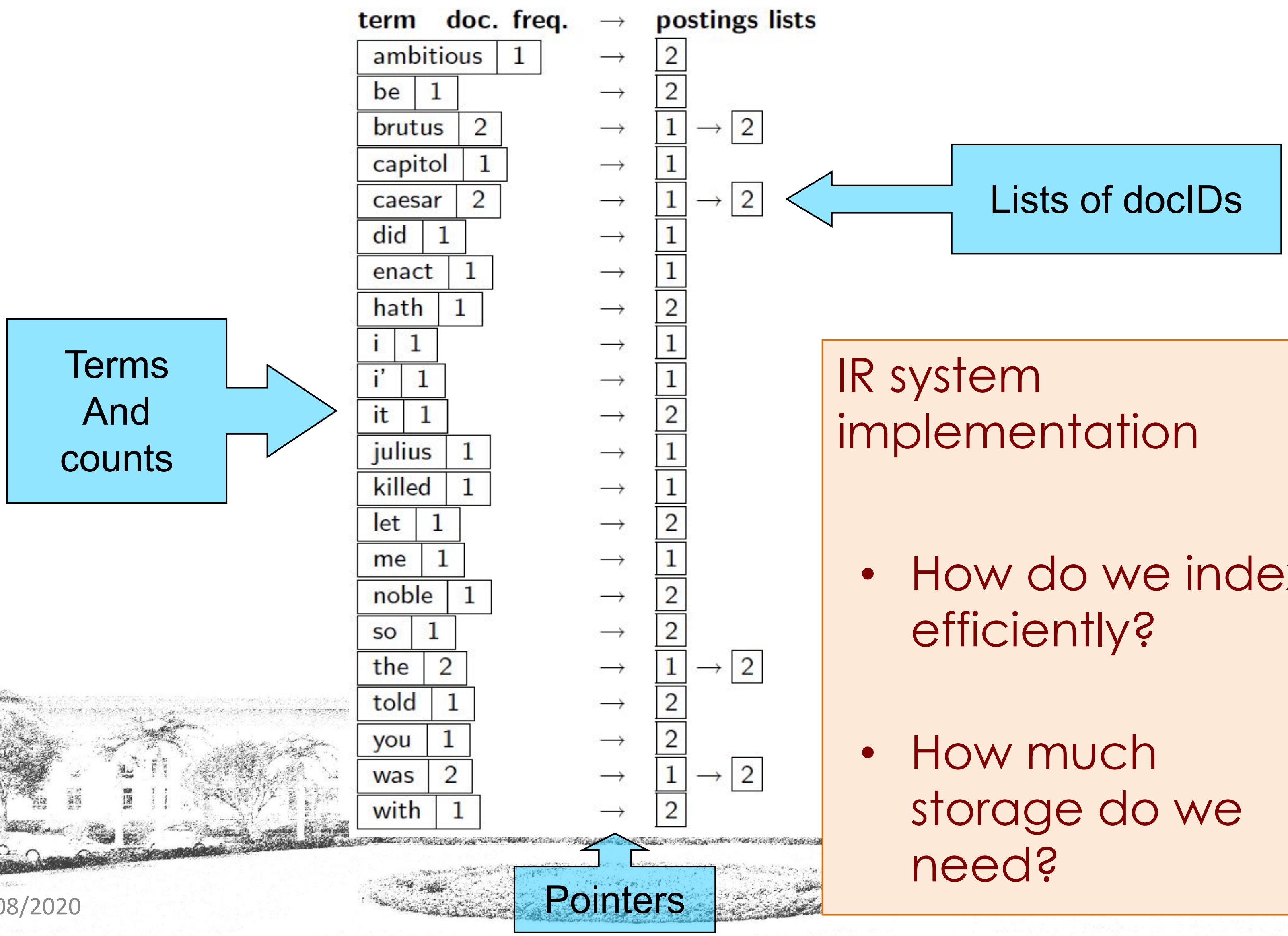
Recap: Creating Inverted Index

d1) Turing machines can define computational processes that do not terminate. The informal definitions of algorithms generally require that the algorithm always terminates. This requirement renders the task of deciding whether a formal procedure is an algorithm impossible in the general case

d2) Typically, when an algorithm is associated with processing information, data can be read from an input source, written to an output device and stored for further processing. Stored data are regarded as part of the internal state of the entity performing the algorithm.

d3) For some such computational process, the algorithm must be rigorously defined: specified in the way it applies in all possible circumstances that could arise. Any conditional steps must be systematically dealt with, case-by-case

Where do we pay in storage?



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
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

IR system implementation

- How do we index efficiently?
- How much storage do we need?

Term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

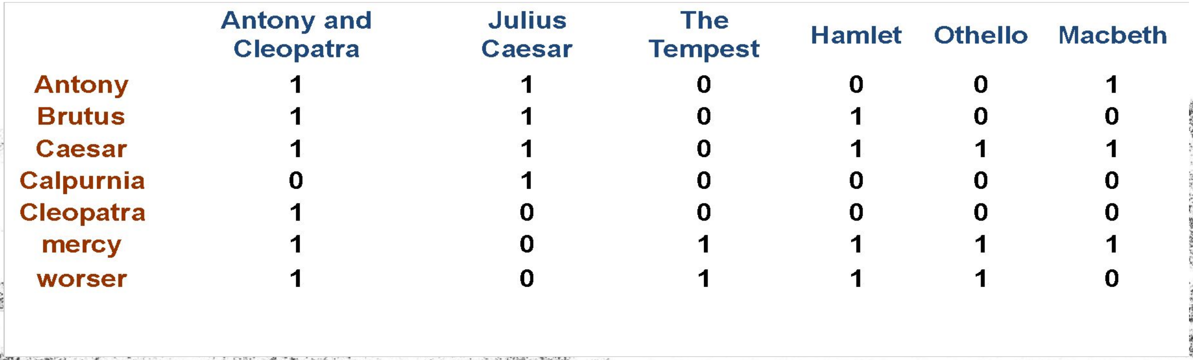
Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

1 if play contains word, 0 otherwise

Incidence vectors

- For each term, we have a vector consisting of 0 / 1
- To answer query: take the vectors for **Brutus**, **Caesar** and **Calpurnia** (complemented) □ bitwise AND
  - 110100 AND
  - 110111 AND
  - 101111 =
  - **100100**

Query:
Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***



	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Bigger collections

- Consider N = 1 million documents, each with about 1000 words
- Average 6 bytes/word including spaces/punctuation
    - ≈ 6GB of data
- Assume that there are M = 500K *distinct* terms among these

Can you build the matrix?

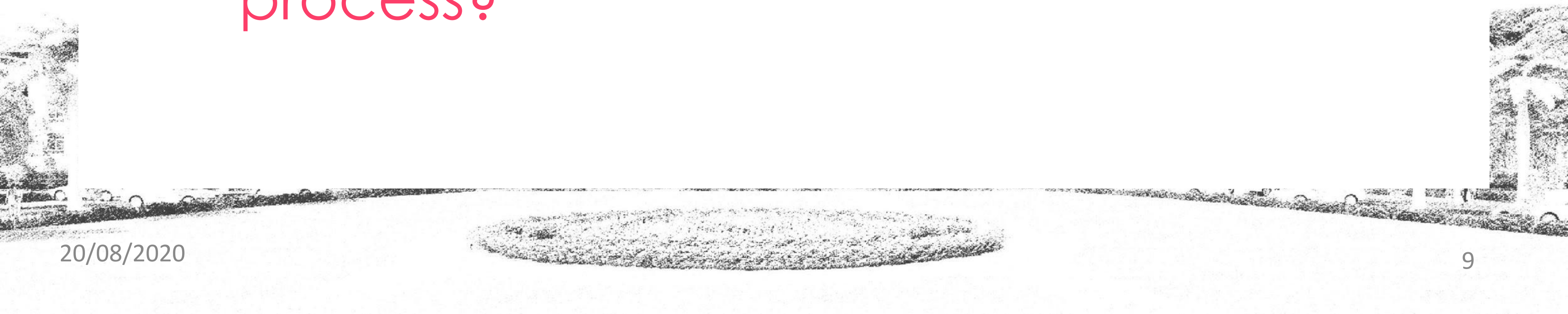
- 500K x 1M matrix has half-a-trillion 0's and 1's.
  - Why??

- But it has no more than one billion 1's.
  – matrix is extremely sparse.

- What's a better representation?
  – We only record the 1 positions.

What is our focus?

- Ask for information
  - Express Information needs in terms of key words
- How do we process a query?
  - Later - what kinds of queries can we process?



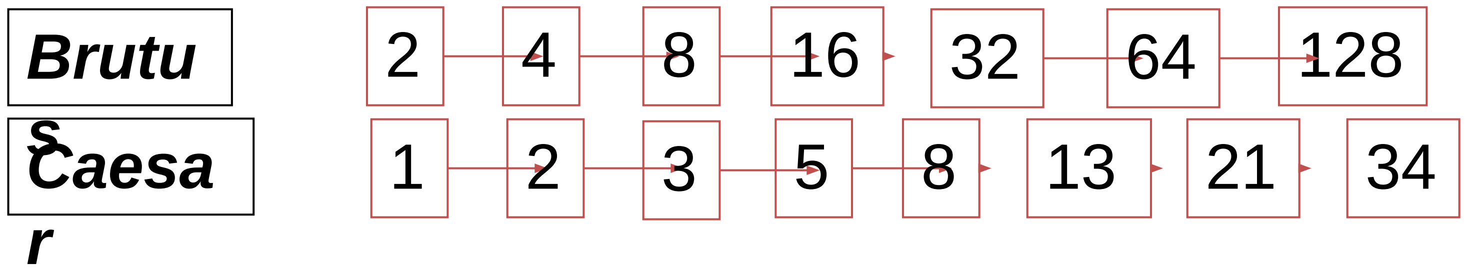
Query processing: AND

- 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):

$$\begin{tikzcd}
\textbf{\textit{Brutus}} & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{8} \arrow[r] & \boxed{16} \arrow[r] & \boxed{32} \arrow[r] & \boxed{64} \arrow[r] & \boxed{128} & \\
\textbf{\textit{Caesar}} & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{3} \arrow[r] & \boxed{5} \arrow[r] & \boxed{8} \arrow[r] & \boxed{13} \arrow[r] & \boxed{21} \arrow[r] & \boxed{34}
\end{tikzcd}$$

Merging of Two Postings List

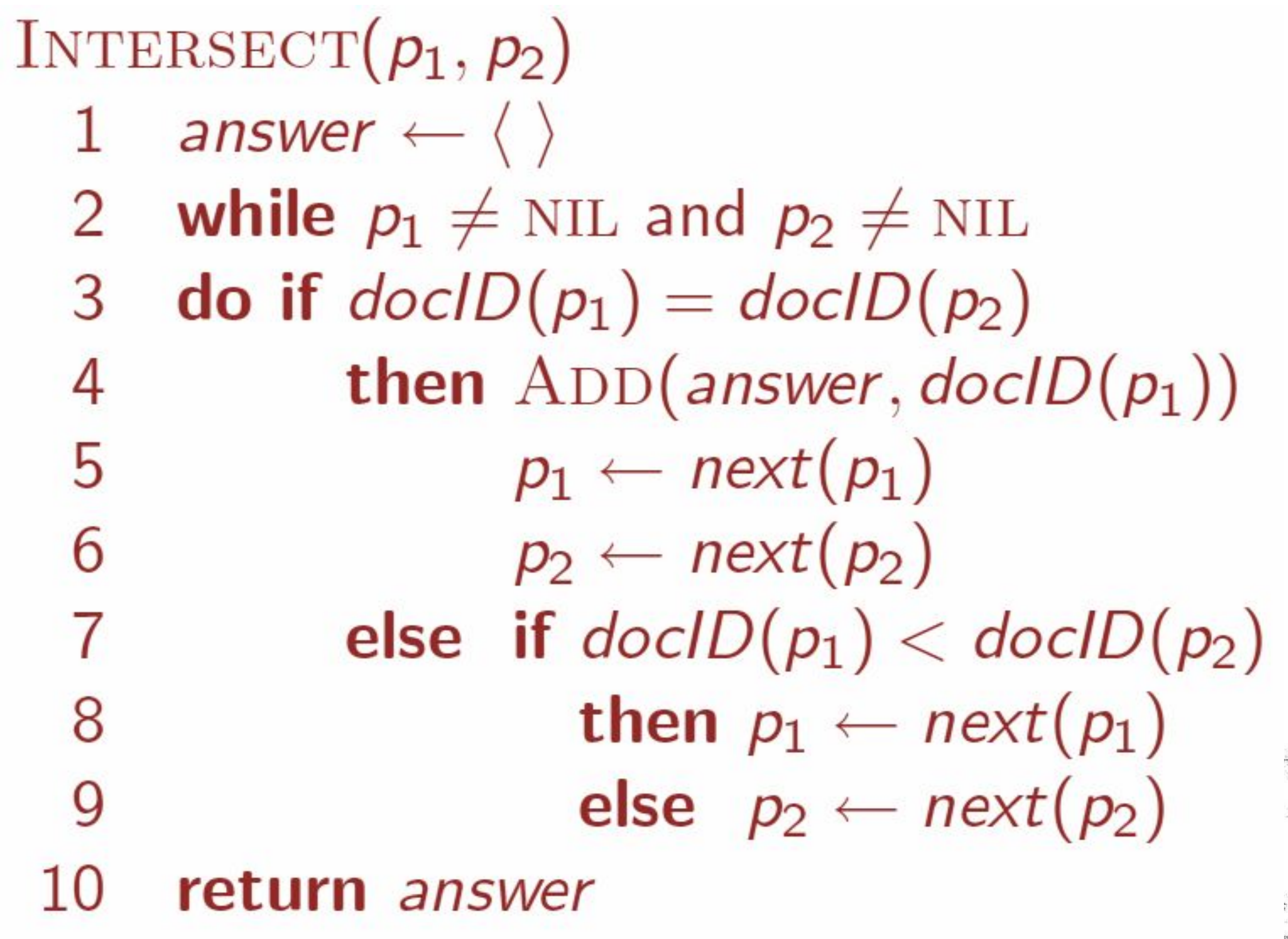
- Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are x and y
the merge takes O(x+y) operations

Crucial: postings sorted by docID.

Intersecting two postings lists (a “merge” algorithm)



```
INTERSECT($p_1, p_2$)
answer $\leftarrow \langle\ \rangle$
while $p_1 \neq$ NIL and $p_2 \neq$ NIL
do if $docID(p_1) = docID(p_2)$
      then ADD($answer, docID(p_1)$)
           $p_1 \leftarrow next(p_1)$
           $p_2 \leftarrow next(p_2)$
      else if $docID(p_1) < docID(p_2)$
              then $p_1 \leftarrow next(p_1)$
              else $p_2 \leftarrow next(p_2)$
return answer
```

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not
  - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades
- Many search systems you still use are Boolean:
  - Email, library catalog, Mac OS X Spotlight

Example: WestLaw http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service
- started in 1975; ranking added in 1992; new federated search added 2010)
- Tens of terabytes of data; ~700,000 users
- Majority of users *still* use **boolean queries**

- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
    - /3 = within 3 words, /S = in same sentence

Example: WestLaw http://www.westlaw.com/

- Another example query:
  - Requirements for disabled people to be able to access a workplace
- Note that SPACE is disjunction, not conjunction!
- Long, precise queries; proximity operators; incrementally developed; not like web search

- Many professional searchers still like Boolean search
  - You know exactly what you are getting

Boolean queries: More general merges

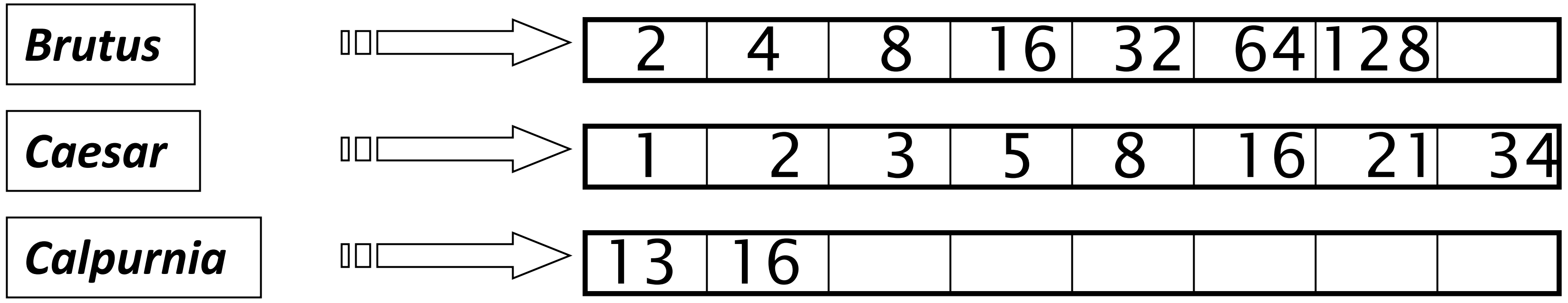
- Exercise: Adapt the merge for the queries:
  - Brutus AND NOT Caesar
  - Brutus OR NOT Caesar
- Can we still run through the merge in time O(x+y)?
  - Linear time?
  - What can we achieve?

Merging

- What about an arbitrary Boolean formula?
- (Brutus OR Caesar) AND NOT
- (Antony OR Cleopatra)
- Can we always merge in “linear” time?
  - Linear in what?
- Can we do better?

Query optimization

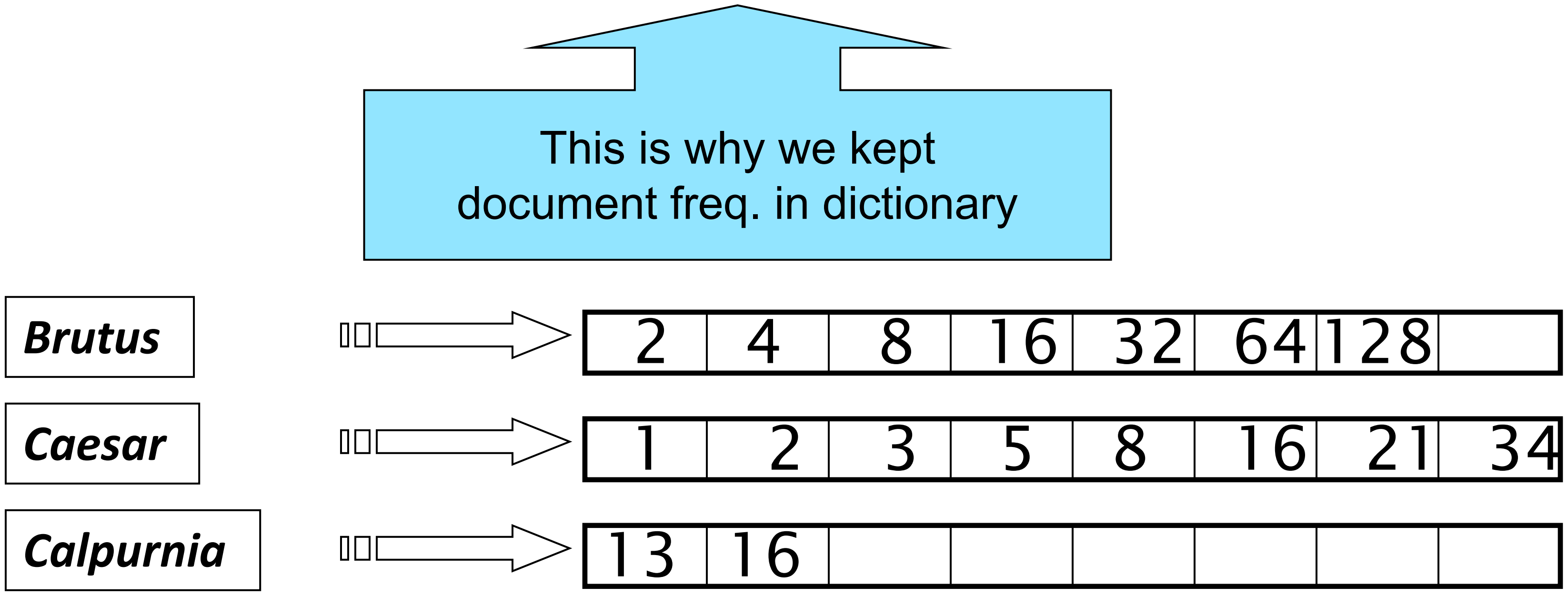
- What is the best order for query processing?
- Consider a query that is an AND of n terms.
- For each of the n terms, get its postings, then AND them together.



Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query optimization example

- Process in order of increasing frequencies:
  - *start with smallest set, then keep cutting further*



Execute the query as (***Calpurnia*** *AND* ***Brutus****)* *AND* ***Caesar***

More general optimization

- e.g., *(**madding** OR **crowd**) AND (**ignoble** OR **strife**)*
- Get doc. freq.'s for all terms
- Estimate the size of each *OR* by the sum of its doc. freq.'s (conservative)
- Process in increasing order of *OR* sizes

Exercise

- Recommend a query processing order for

(tangerine** OR **trees)** AND **(marmalade** OR **skies)** AND **(kaleidoscope** OR **eyes)

- Which two terms should we process first?

Term	Freq
eyes	213312
kaleidosco	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Query Processing - Exercises

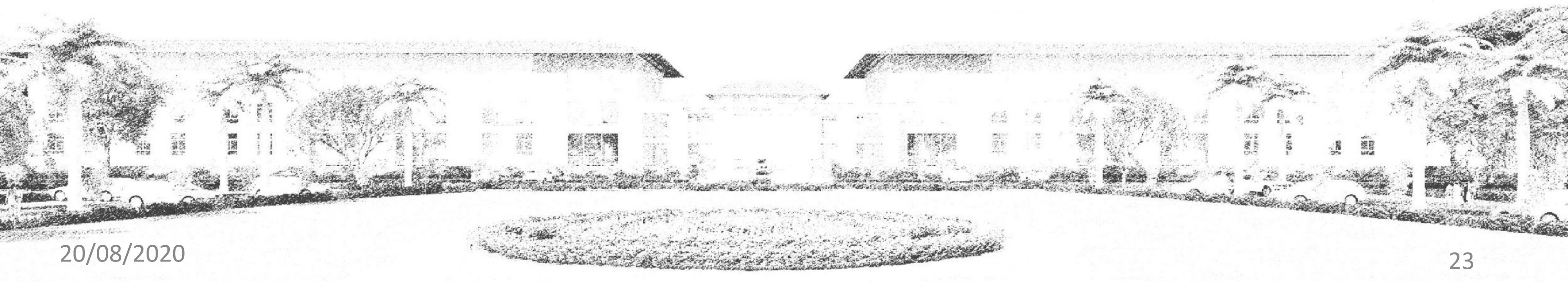
- Exercise: If the query is ***friends*** *AND* ***romans*** *AND (NOT* ***countrymen****)*, how could we use the freq of ***countrymen***?

- Exercise: Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?

- Hint: Begin with the case of a Boolean *formula* query: in this, each query term appears only once in the query

Exercise

- Try the search feature at http://www.rhymezone.com/shakespeare/
- Write down five search features you think it could do better



Summary

In this class, we focused on:

(a) Words / Terms / Lexical Units

(b) Preparing Term – Document matrix

(c) Boolean Retrieval

(d) Inverted Index Construction

   i. Computational Cost

   ii. Managing Bigger Collections

   iii. How much storage is required?

   iv. Boolean Queries: Exact match

Assistance

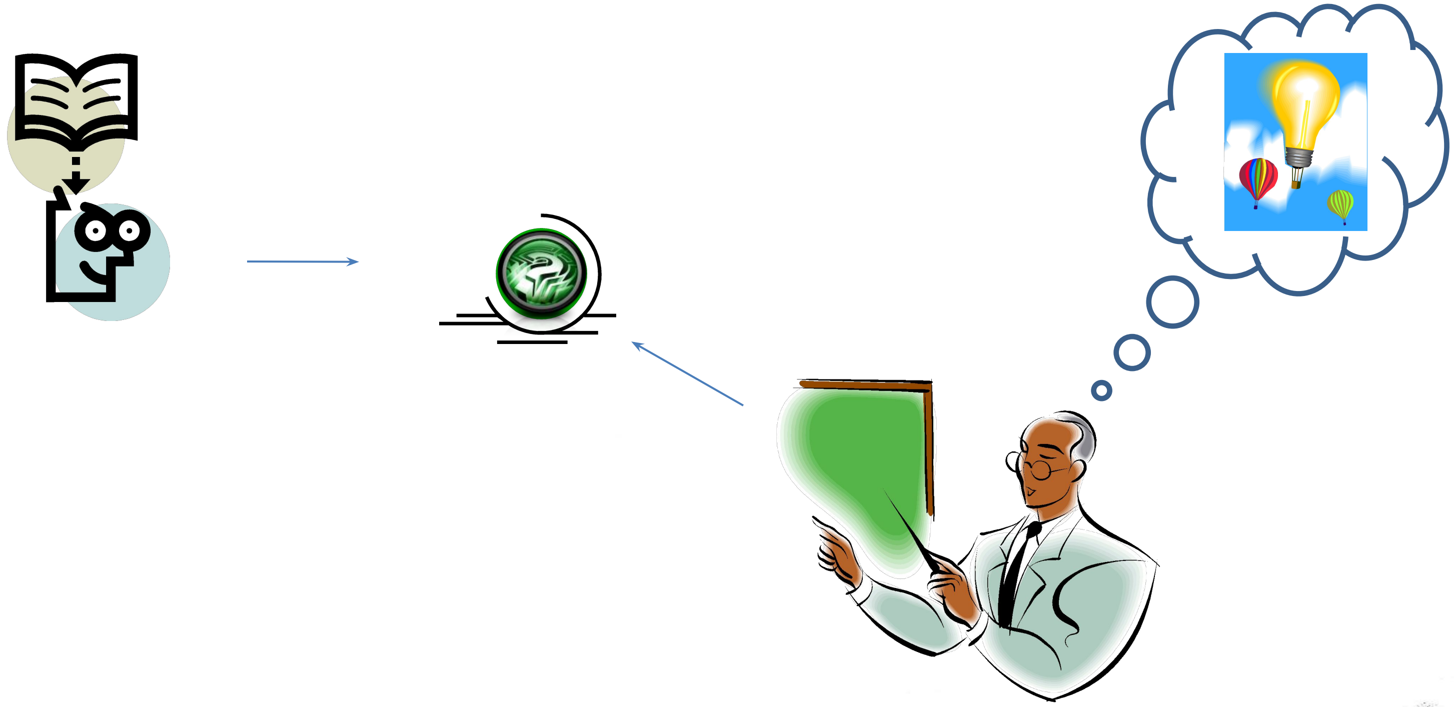
- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
 (the best way to reach me faster)

Acknowledgements

Thanks to ALL RESEARCHERS:

1. Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.
2. Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
3. Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
4. Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
5. Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
6. Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)

Thanks …



… Questions ???