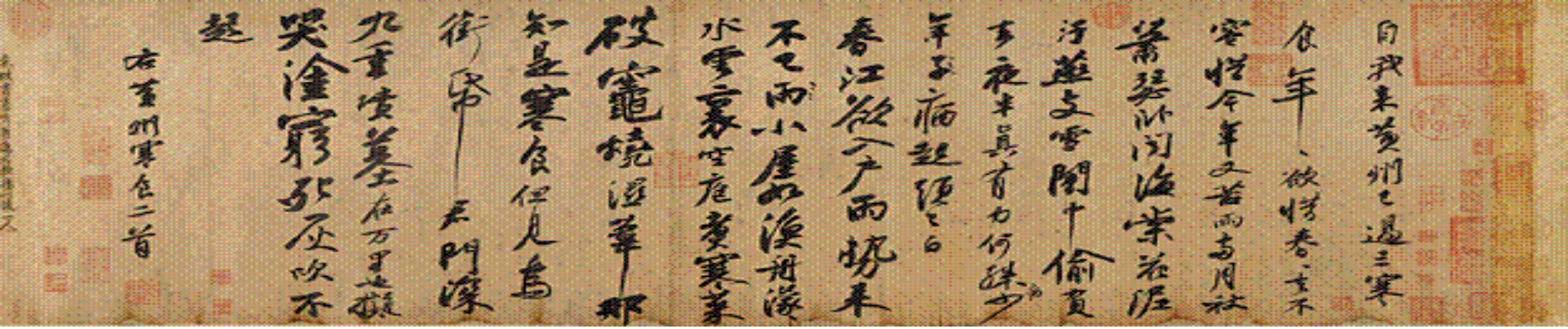


信息检索
Information Retrieval

教师：孙茂松

Tel:62781286

Email:sms@tsinghua.edu.cn

TA：**胡锦毅**

Email:hu-jy21@mails.tsinghua.edu.cn

郑重声明

- 此课件仅供选修清华大学计算机系本科生课《信息检索》(40240372)的学生个人学习使用，所以只允许学生将其下载、存贮在自己的电脑中。未经孙茂松本人同意，任何人不得以任何方式扩散之。否则，由此可能引起的一切涉及知识产权的法律责任，概由该人负责。
- 此课件仅限孙茂松本人讲课使用。除孙茂松本人外，凡授课过程中，PTT文件显示此《郑重声明》之情形，即为侵权使用。

第二章 信息检索系统的基本框架(Part 1)

2.1 信息检索基本模型

- What is information?

"something that (1) is represented by a set of symbols, (2) has some structure, and (3) can be read and to some extent understood by users of information " (Meadow)

- Natural language texts： The term "unstructured data" refers to data which does not have clear, semantically overt, easy-for-a-computer structure.

2.1 信息检索基本模型

- IR is concerned with the representation, storage, organization and accessing of information items.

- For a given information problem, the purpose of the system is to capture wanted items and to filter out unwanted items.

2.1 信息检索基本模型

- Definition for Information Retrieval

Salton (1989): "Information-retrieval systems process files of records and requests for information, and identify and retrieve from the files certain records in response to the information requests. The retrieval of particular records depends on the similarity between the records and the queries, which in turn is measured by comparing the values of certain attributes to records and information requests."

Kowalski (1997): "An Information Retrieval System is a system that is capable of storage, retrieval, and maintenance of information. Information in this context can be composed of text, images, audio, video, and other multi-media objects."

2.1 信息检索基本模型

What are the components of an IR system?

* Document processing (indexing)
* Query input
* Document-query “matching”
* Output module
* Feedback module
* User interface

2.2 IR基本文件结构

Direct file vs. inverted file

		Related information items				
Topic	Computer	1		3		
	Information	1	2		4	
	Retrieval	1	2		4	5
	Systems	1		3	4	5
	Users		2			5

(a)

Item number	1	2	3	4	5
Author	Ash	Brown	Jones	Reynolds	Smith
Title	Aspects of Computerized Information Retrieval Systems	A Survey of Users of Information Retrieval	The History of Computer Systems	The State of the Art of Information Retrieval Systems	Users of New Retrieval Systems
Topic	Computer Information Retrieval Systems	Information Retrieval Users	Computer Systems	Information Retrieval Systems	Retrieval Systems Users

(b)

Figure 1-10 Sample inverted file organization. (a) Inverted index identifying item numbers corresponding to particular topics. (b) Sample information items.

2.2 IR基本文件结构

		Related information items					
Topic	Computer	1		3			6
	Information	1	2		4		6
	Retrieval	1	2		4	5	
	Systems	1		3	4	5	6
	Users		2			5	

(a)

Item number	1	2	3	4	5	6
Author	Ash	Brown	Jones	Reynolds	Smith	David
Title	Aspects of Computerized Information Retrieval Systems	A Survey of Users of Information Retrieval	The History of Computer Systems	The State of the Art of Information Retrieval Systems	Users of New Retrieval Systems	A Study of Computerized Information Systems
Topic	Computer Information Retrieval Systems	Information Retrieval Users	Computer Systems	Information Retrieval Systems	Retrieval Systems Users	Computer Information System

(b)

Figure 1-11 Inverted file with added item. (a) Inverted index with a new item 6. (b) Sample information items with added item 6.

An index to the index may be further built.

2.2 IR基本文件结构

Information items

Topics	Document 1	Document 2	Document 3
Term 1	1	0	1
Term 2	1	1	0
Term 3	0	1	1
Term 4	1	1	1

(a)

Topics

Information items	Term 1	Term 2	Term 3	Term 4
Document 1	1	1	0	1
Document 2	0	1	1	1
Document 3	1	0	1	1

(b)

Figure 1-12 Inverted and direct file examples. (a) Inverted file. (b) Direct file.

2.3 针对倒排文件的基本操作

- 布尔检索（Boolean retrieval）

- 布尔表达式

  AND, OR, NOT　　XOR

  set intersection, set union, set difference

(APPLE AND ORANGE) OR (BANANA AND ORANGE)

分析：逆波兰式

2.4 对倒排文件的进一步考察

Query: Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?

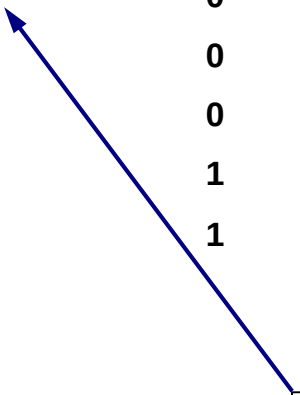
Could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar*** but not containing ***Calpurnia***?

Slow (for large corpora)

Term-document incidence

To answer query: take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** bitwise *AND*: 110100 *AND* 110111 *AND* 101111 = 100100.

	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



1 if play contains word, 0 otherwise

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

Answers to query

- Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius ***Caesar*** dead,
He cried almost to roaring; and he wept
When at Philippi he found ***Brutus*** slain.

- Hamlet, Act III, Scene ii

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

2.4 对倒排文件的进一步考察

问题：Sparse Matrix

- Consider *n* = 1M documents, each with about 1K words.
- Avg 6 bytes/word incl spaces/punctuation
  - 6GB of data.
- Say there are *m* = 500K *distinct* words among these.
- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's.
  - matrix is extremely sparse.
- What's a better representation?

2.4 对倒排文件的进一步考察

Documents are parsed to extract words and these are saved with the Document ID.

Doc 1

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

Doc 2

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

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

2.4 对倒排文件的进一步考察

After all documents have been parsed, the inverted file is sorted by terms

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

2.4 对倒排文件的进一步考察

Multiple term entries in a single document are merged and frequency information added

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

2.4 对倒排文件的进一步考察

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Term	N docs	Tot Freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

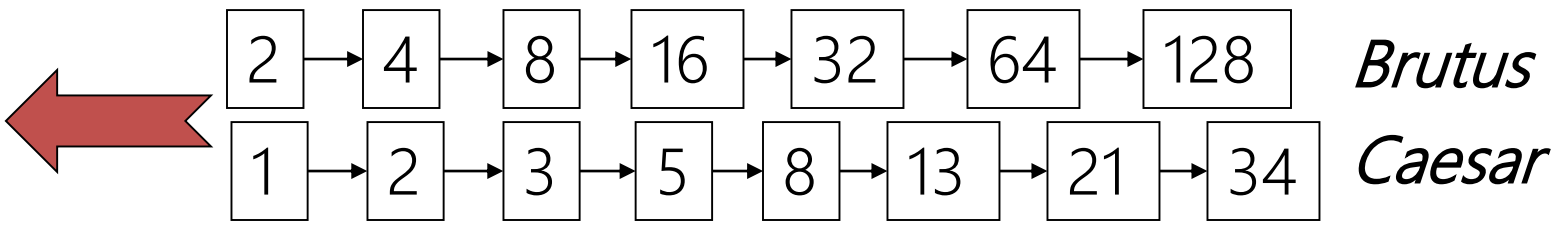
The file is commonly split into a *Dictionary* and a *Postings list* file

Query processing: AND

- Consider processing the query:

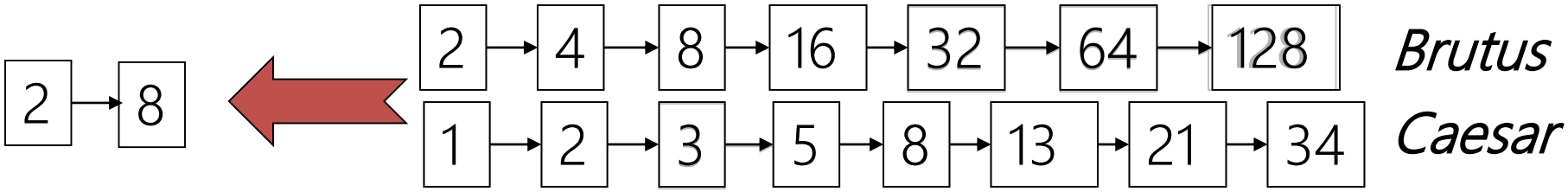
  Brutus *AND* ***Caesar***

  - Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.
  - Locate *Caesar* in the Dictionary;
    - Retrieve its postings.
  - "Merge" the two postings:



The merge

- 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.

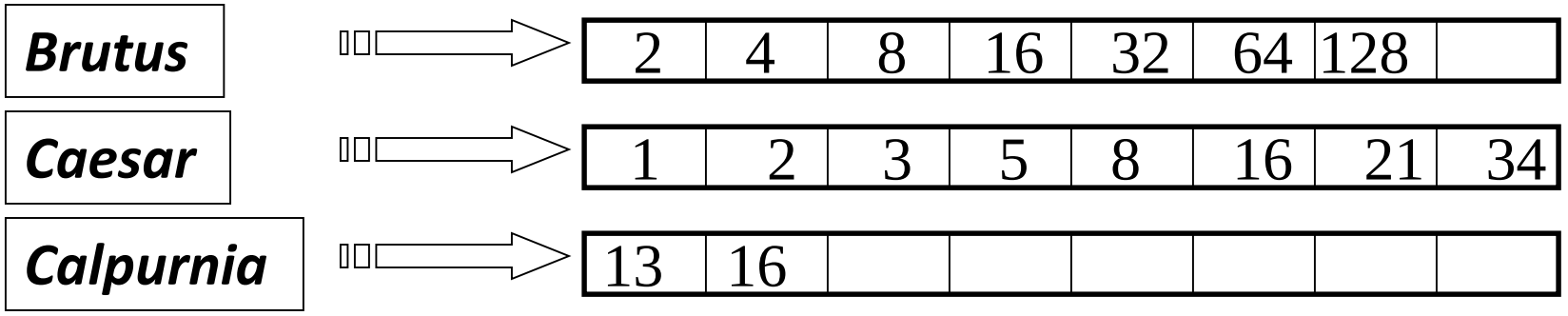
Crucial: postings sorted by docID.

Intersecting two postings lists (a "merge" algorithm)

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Query optimization

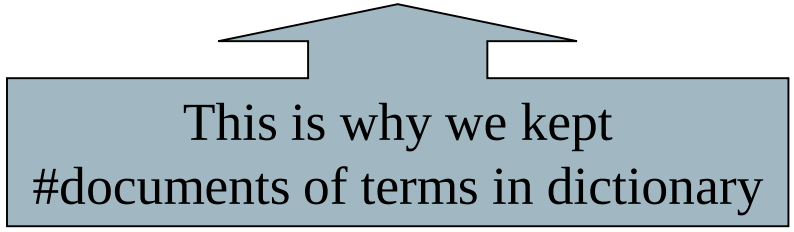
- What is the best order for query processing?
- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.

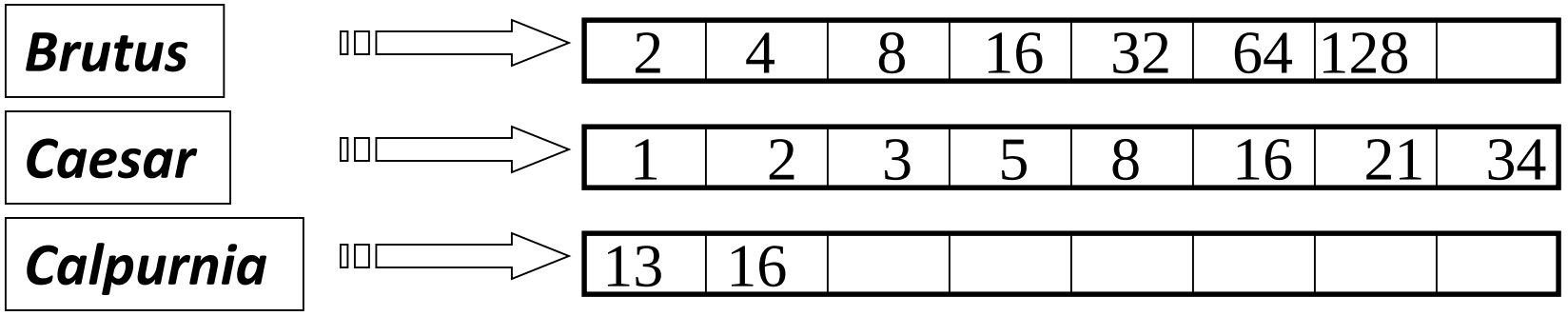


Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query optimization

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





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

Query optimization

```
INTERSECT(⟨t_1, ..., t_n⟩)
1  terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
2  result ← postings(first(terms))
3  terms ← rest(terms)
4  while terms ≠ NIL and result ≠ NIL
5  do result ← INTERSECT(result, postings(first(terms)))
6     terms ← rest(terms)
7  return result
```

- *(**madding** OR **crowd**) AND (**ignoble** OR **strife**)*

  *AND (**increasing** OR **conservative**)*
- Get posting sizes for all terms.
- Estimate the size of each *OR* by the sum of its postings sizes (#documents) of related terms.
- 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)

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

Boolean queries: More general merges

- Question:

  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*)?

What can we achieve?

Boolean queries: More general merges

What about an arbitrary Boolean formula?

***(Brutus** OR **Caesar)** AND NOT*
(Antony** OR **Cleopatra)

- Can we always merge in "linear" time?
  - Linear in what?