Introduction to

Information Retrieval

cs160

Introduction

David Kauchak

adapted from:

http://www.stanford.edu/class/cs276/handouts/lecture1-intro.ppt

Introductions

- Name/nickname
- Dept., college and year
- One interesting thing about yourself
- Why are you taking this class?
- What topics/material would you like to see covered?
- Plans after graduation

Administrative

- Web page: www.cs.pomona.edu/classes/cs160/
- Syllabus…
- Administrative handout…
- Class feedback
- In class participation
- Workload

- Homework 1 available soon
- Programming assignment 1 available soon
  - Due time?

Information retrieval (IR)

- What comes to mind when I say "information retrieval"?

- Where have you seen IR? What are some real-world examples/uses?
  - Search engines
  - File search (e.g. OS X Spotlight, Windows Instant Search, Google Desktop)
  - Databases?
  - Catalog search (e.g. library)
  - Intranet search (i.e. corporate networks)

Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content
  - Find all documents about computer science
  - Find all course web pages at Pomona
  - What is the cheapest flight from LA to NY?
  - Who is was the 15th president?

Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

What is the difference between an *information need* and a *query?*

Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

Information need	Query
• Find all documents about computer science	"computer science"
• Find all course web pages at Pomona	Pomona AND college AND *url-contains* class
• Who is was the 15th president?	WHO=president NUMBER=15

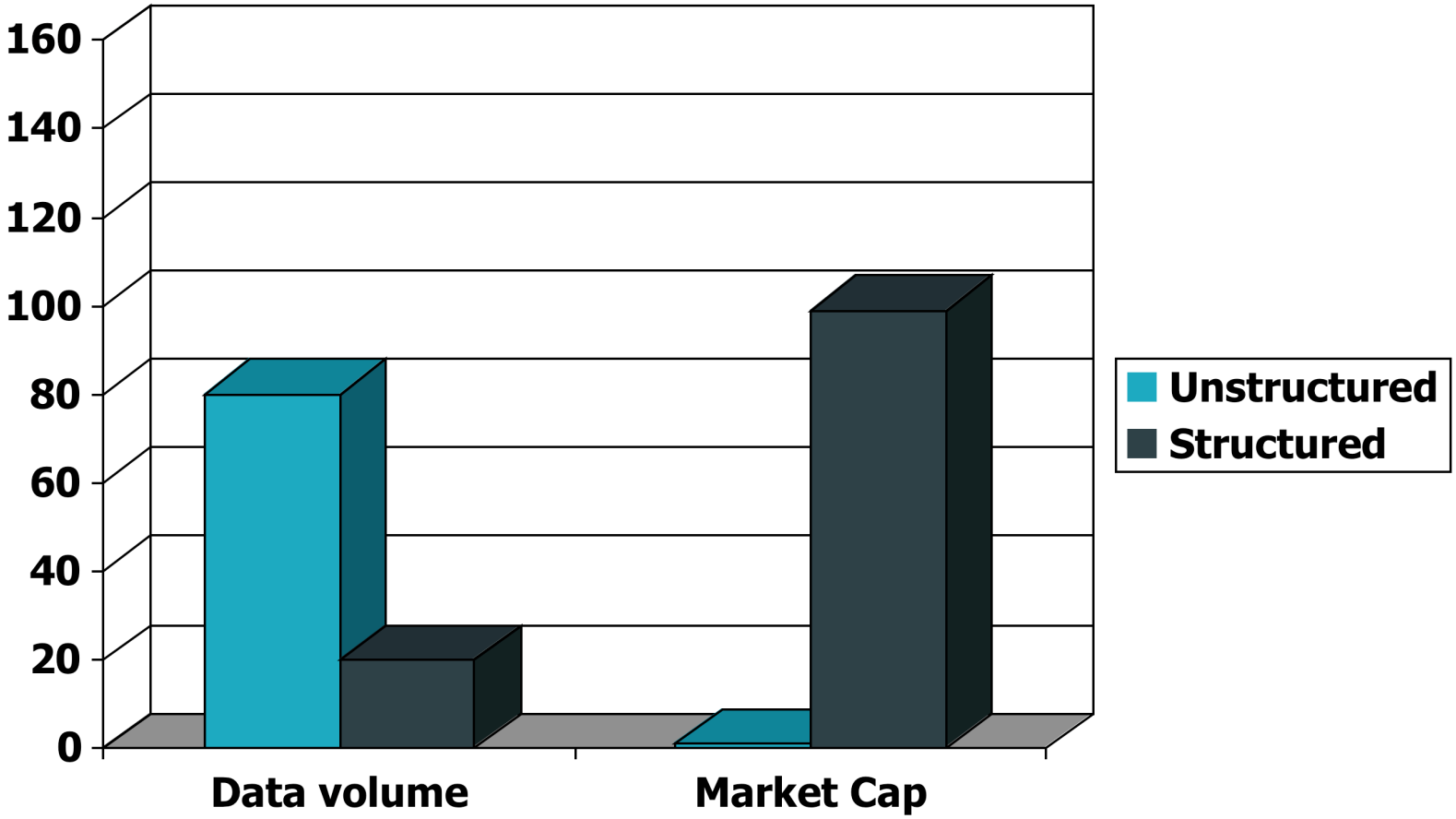
IR vs. databases

- Structured data tends to refer to information in "tables"

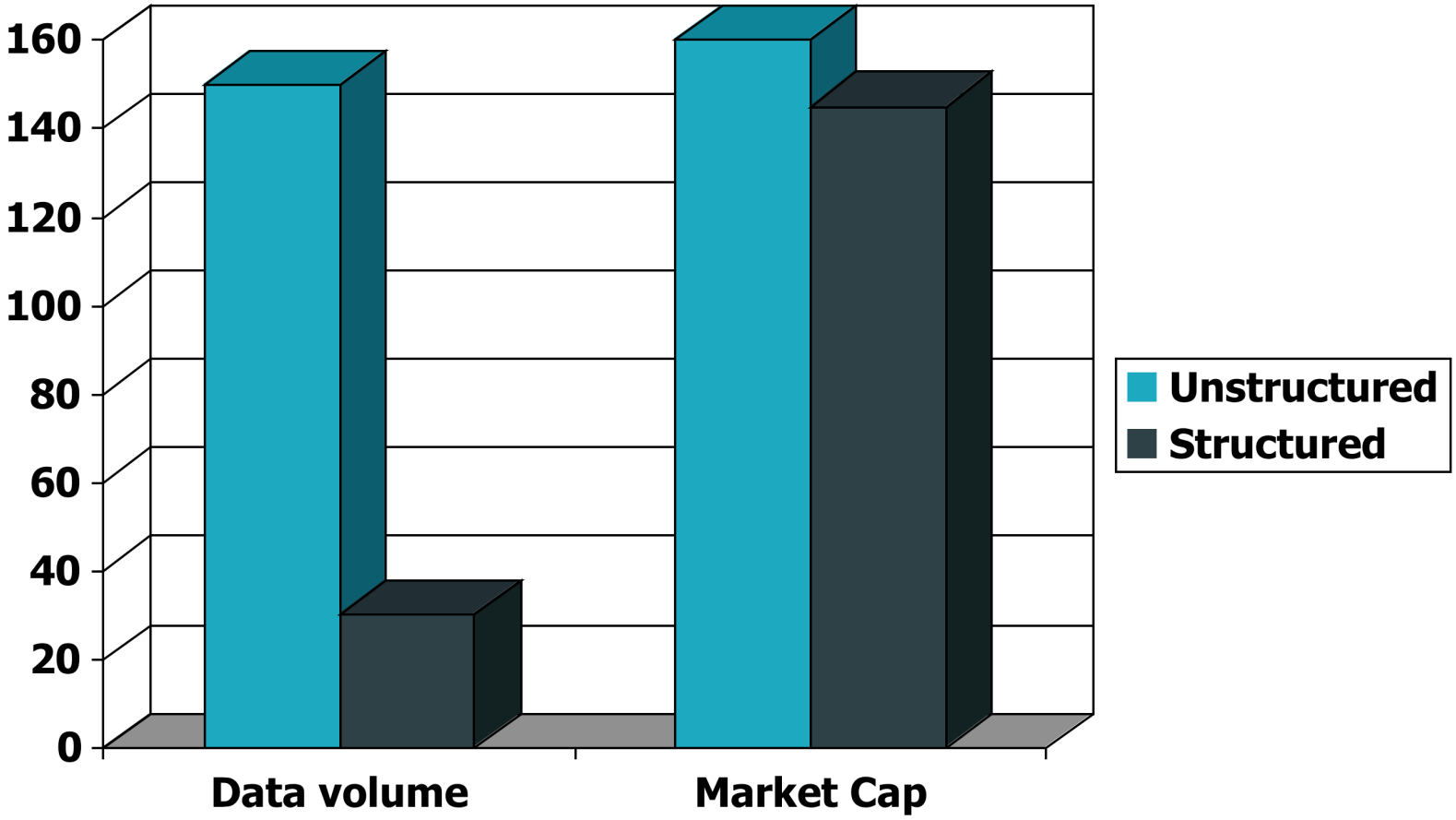
Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows numerical range and exact match (for text) queries, e.g.,
Salary < 60000 AND Manager = Smith.

Unstructured (text) vs. structured (database) data in 1996



Unstructured (text) vs. structured (database) data in 2006



Unstructured data in 1680

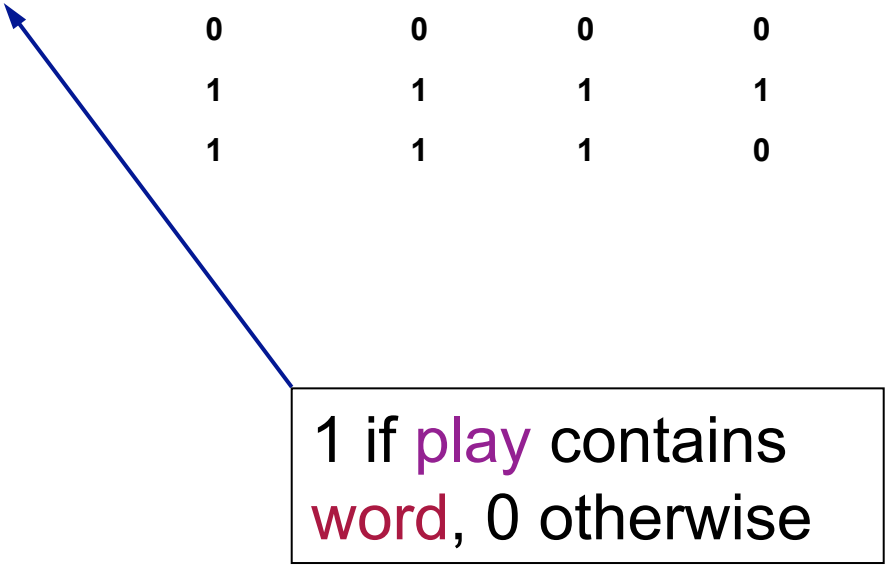
- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out plays containing ***Calpurnia***. Any problems with this?
  - Slow (for large corpora)
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Ranked retrieval (best documents to return)
    - Later lectures

Unstructured data in 1680

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- How might we speed up this type of query?
- Indexing: for each word, keep track of which documents it occurs in

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



Incidence vectors

- For each term, we have a 0/1 vector
  - Caeser = 110111
  - Brutus = 110100
  - Calpurnia = 010000

	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

Incidence vectors

- For each term, we have a 0/1 vector
  - Caeser = 110111
  - Brutus = 110100
  - Calpurnia = 010000

How can we get the answer from these vectors?

Incidence vectors

- For each term, we have a 0/1 vector
  - Caeser = 110111
  - Brutus = 110100
  - Calpurnia = 010000
- Bitwise AND the vectors together using the complemented vector for all NOT queries
- Caeser AND Brutus AND COMPLEMENT(Calpurnia)
  - 110111 & 110100 & ~010000 =
  - 110111 & 110100 & 101111 =
  - 100100

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.

Incidence vectors

- For each term, we have a 0/1 vector
  - Caeser = 110111
  - Brutus = 110100
  - Calpurnia = 010000
- Bitwise AND the vectors together using the complemented vector for all NOT queries

Any problem with this approach?

Bigger collections

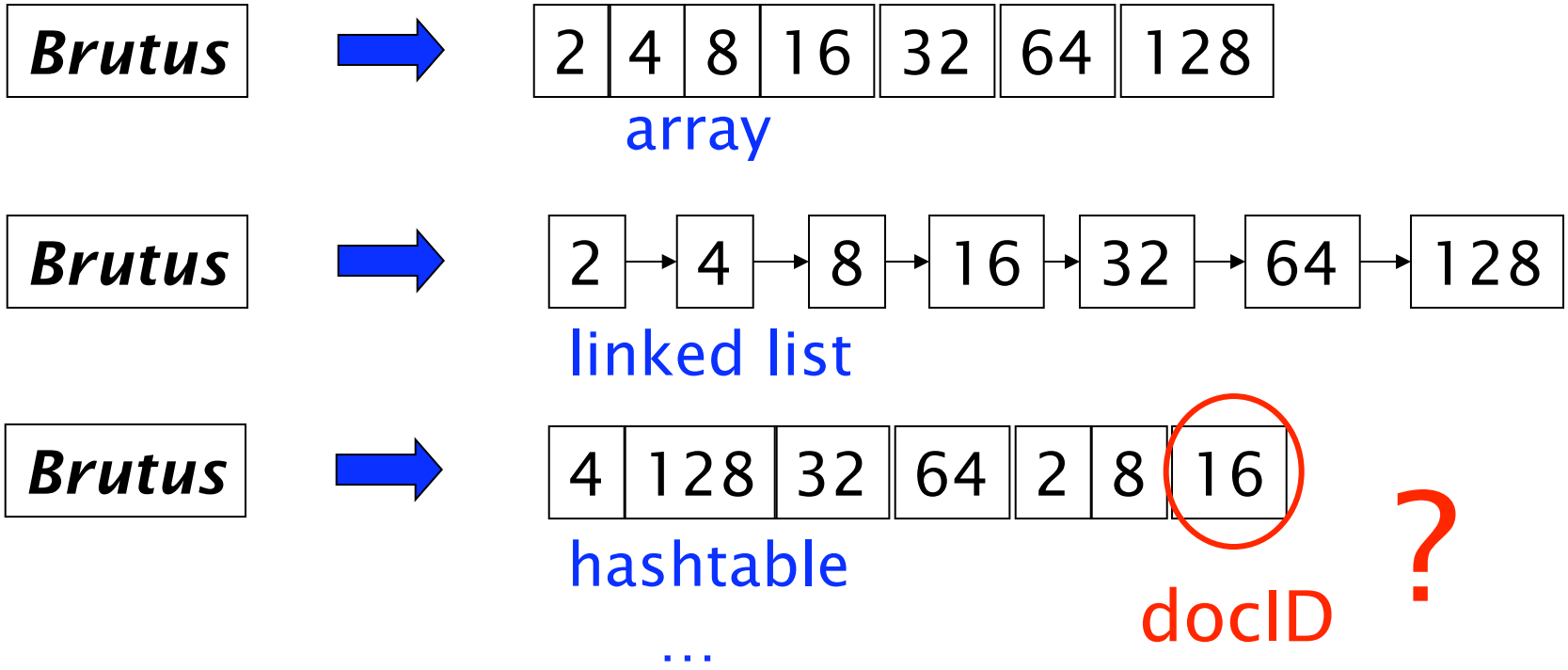
- Consider *N* = 1 million documents, each with about 1000 words
- Say there are *M* = 500K *distinct* terms among these. How big is the incidence matrix?
- The matrix is a 500K by 1 million matrix = half a trillion 0's and 1's
  - Even for a moderate sized data set we can't store the matrix in memory
- Each vector has 1 million entries
  - Bitwise operations become much more expensive

What does the matrix look like?

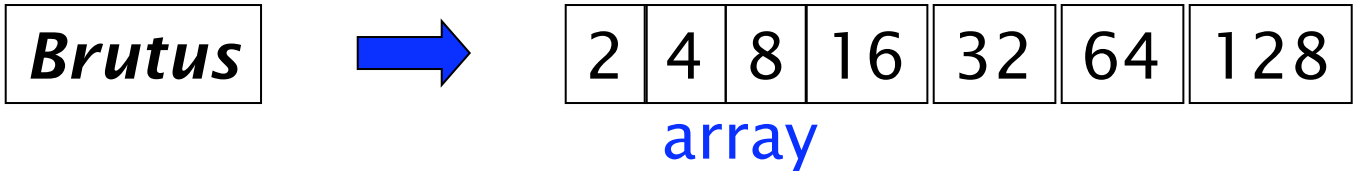
- Consider *N* = 1 million documents, each with about 1000 words
- Extremely sparse!
- How many 1's does the matrix contain?
  - no more than one billion
  - Each of the 1 million documents has at most 1000 1's
  - In practice, we'll see that the number of unique words in a document is much less than this
- What's a better representation?
  - Only record the 1 positions

Inverted index

- For each term, we store a list of all documents that contain it
- What data structures might we use for this?

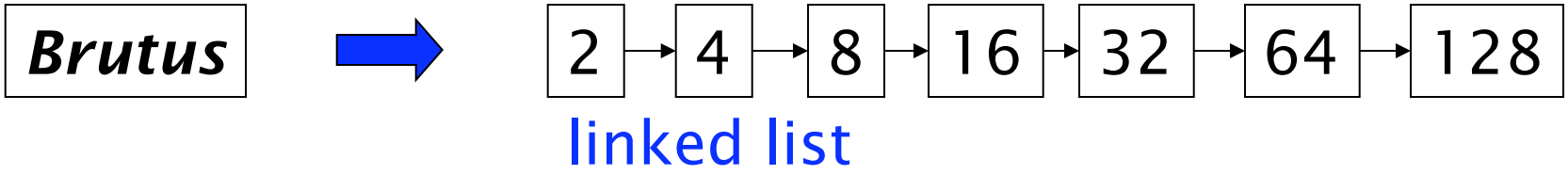


Inverted index representation



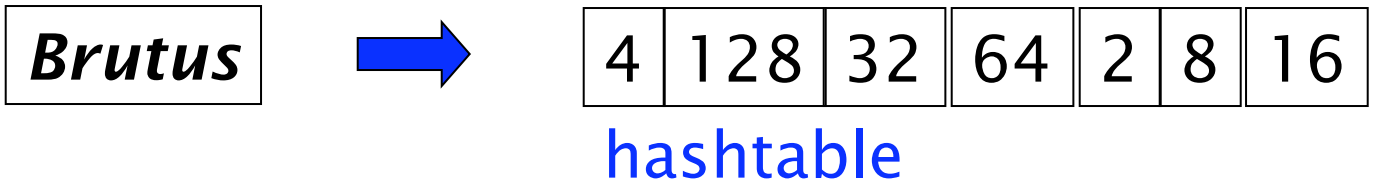
- Pros
  - Simple to implement
  - No extra pointers required for data structure
  - Contiguous memory
- Cons
  - How do we pick the size of the array?
  - What if we want to add additional documents?

Inverted index representation



- Pros
  - Dynamic space allocation
  - Insertion of new documents is straightforward
- Cons
  - Memory overhead of pointers
  - Noncontiguous memory access

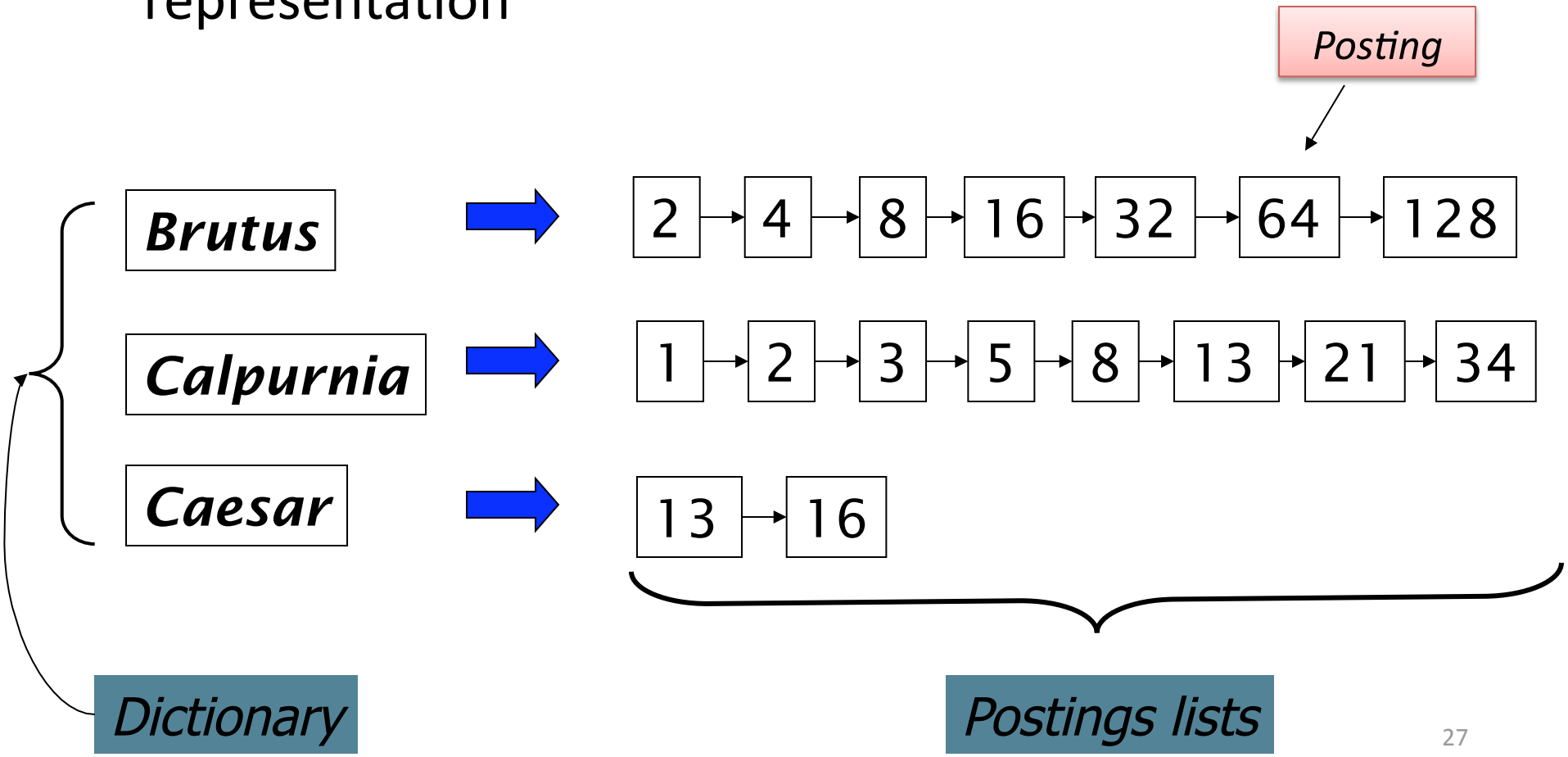
Inverted index representation



- Pros
  - Search in constant time
  - Contiguous memory
- Cons
  - How do we pick the size?
  - What if we want to add additional documents?
  - May have to rehash if we increase in size
  - To get constant time operations, lots of unused slots/memory

Inverted index

- The most common approach is to use a linked list representation



Inverted index construction

Documents to be indexed

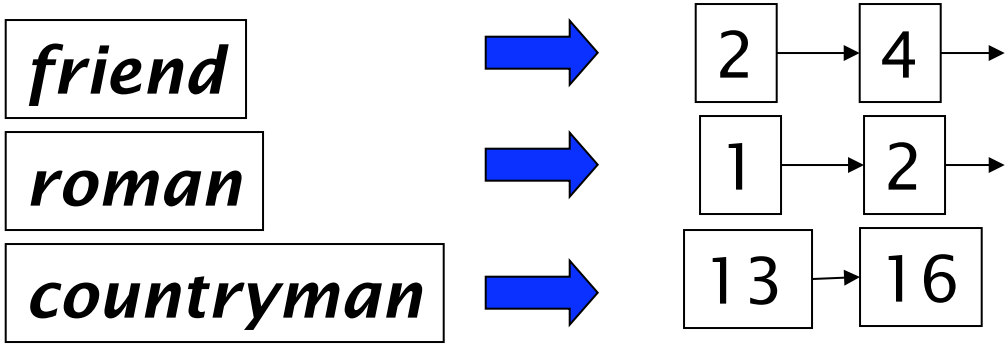
Friends, Romans, countrymen.

text preprocessing

friend , roman , countrymen .

indexer

Inverted index



Boolean retrieval

- In the boolean retrieval model we ask a query that is a boolean expression:
  - A boolean query uses *AND*, *OR* and *NOT* to join query terms
    - Caesar *AND* Brutus *AND NOT* Calpurnia
    - Pomona *AND* College
    - (Mike *OR* Michael) *AND* Jordan *AND NOT*(Nike *OR* Gatorade)
- Given only these operations, what types of questions can't we answer?
  - Phrases, e.g. "Pomona College"
  - Proximity, "Michael" within 2 words of "Jordan"

Boolean retrieval

- Primary commercial retrieval tool for 3 decades
- Professional searchers (e.g., lawyers) still like boolean queries
- Why?
  - You know exactly what you're getting, a query either matches or it doesn't
  - Through trial and error, can frequently fine tune the query appropriately
  - Don't have to worry about underlying heuristics (e.g. PageRank, term weightings, synonym, etc…)

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

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- 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
  - All words starting with "LIMIT"

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

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- 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

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

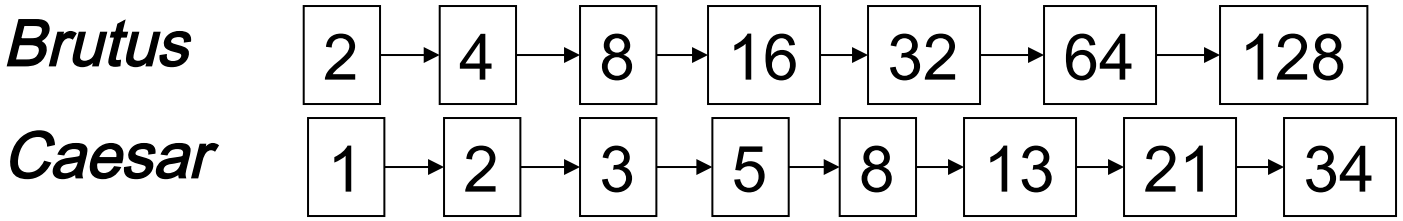
- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of terabytes of data; 700,000 users
- Majority of users *still* use boolean queries
- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /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
  - disabl! /p acces\s! /s work-site work-place (employment /3 place)
- Long, precise queries; proximity operators; incrementally developed; not like web search
- Professional searchers often like Boolean search:
  - Precision, transparency and control
- But that doesn't mean they actually work better….

Query processing: AND

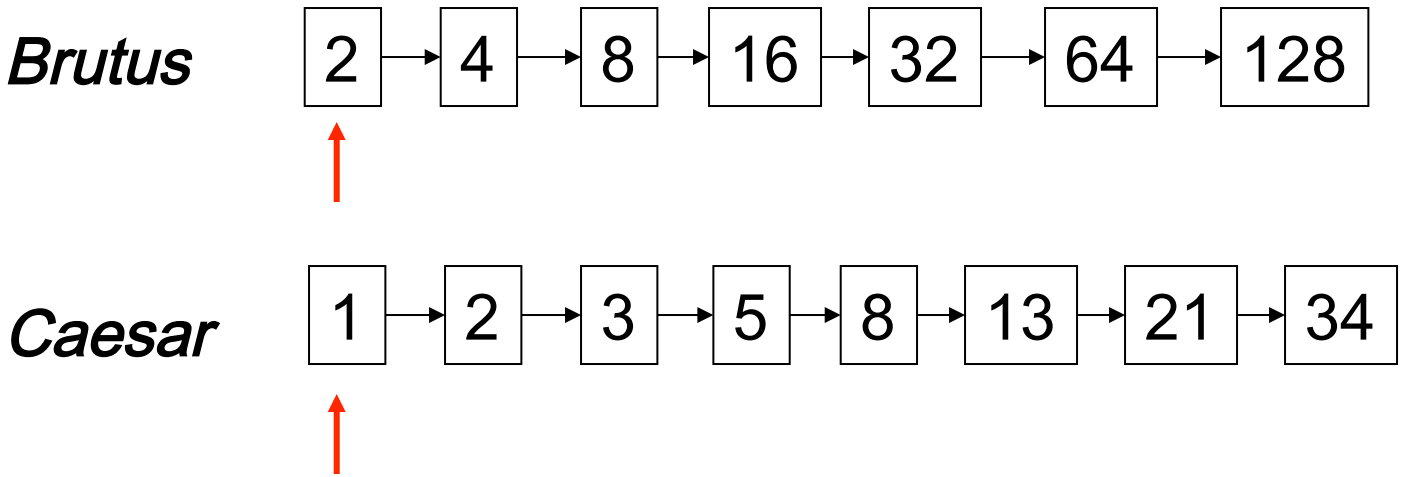
- What needs to happen to process:
  Brutus *AND* ***Caesar***
- Locate ***Brutus*** and ***Caesar*** in the Dictionary;
  - Retrieve postings lists



- "Merge" the two postings:

The merge

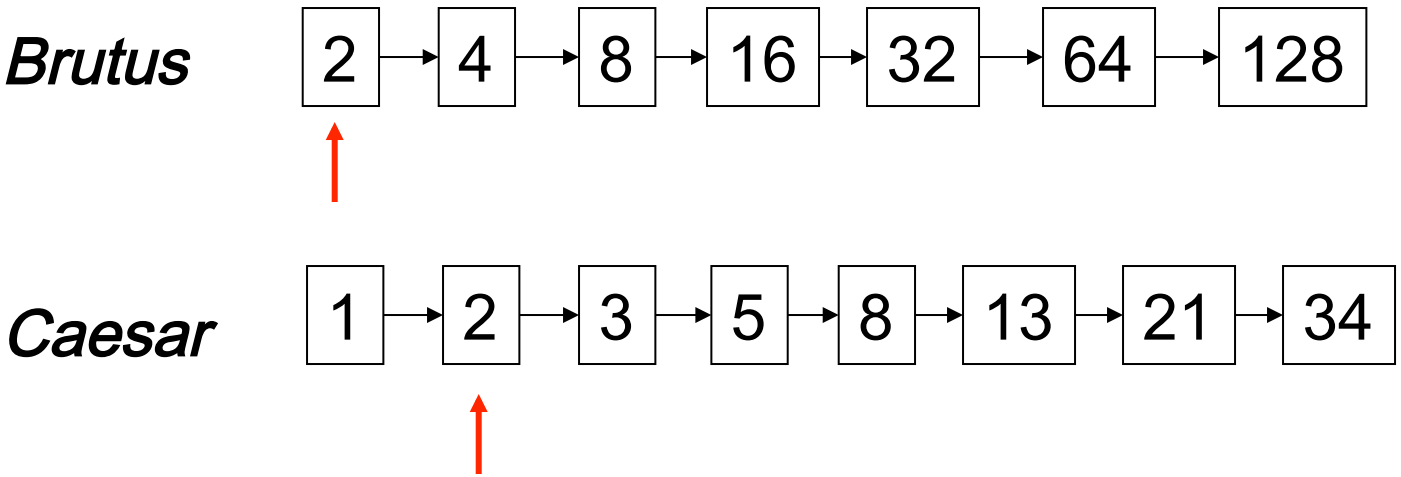
- Walk through the two postings simultaneously



Brutus AND Caesar

The merge

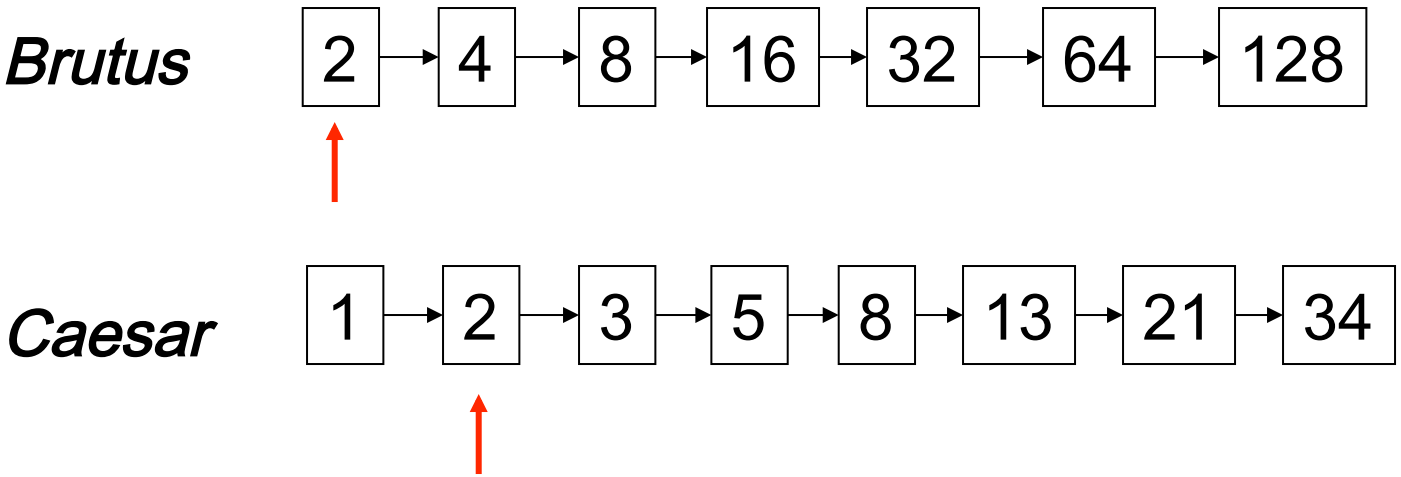
- Walk through the two postings simultaneously



Brutus AND Caesar

The merge

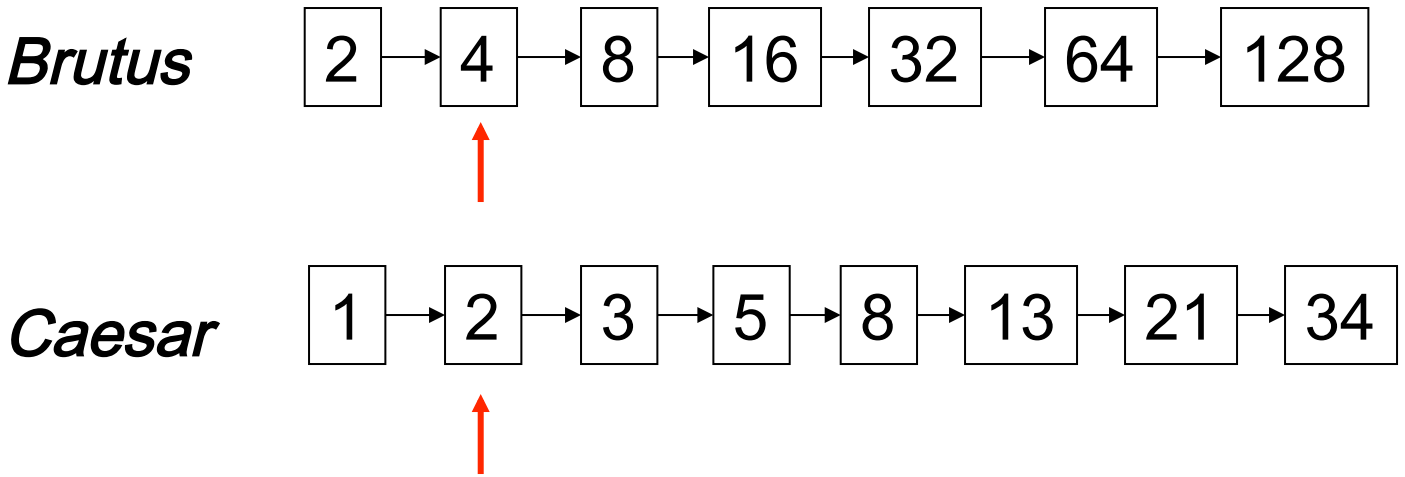
- Walk through the two postings simultaneously



Brutus AND Caesar 2

The merge

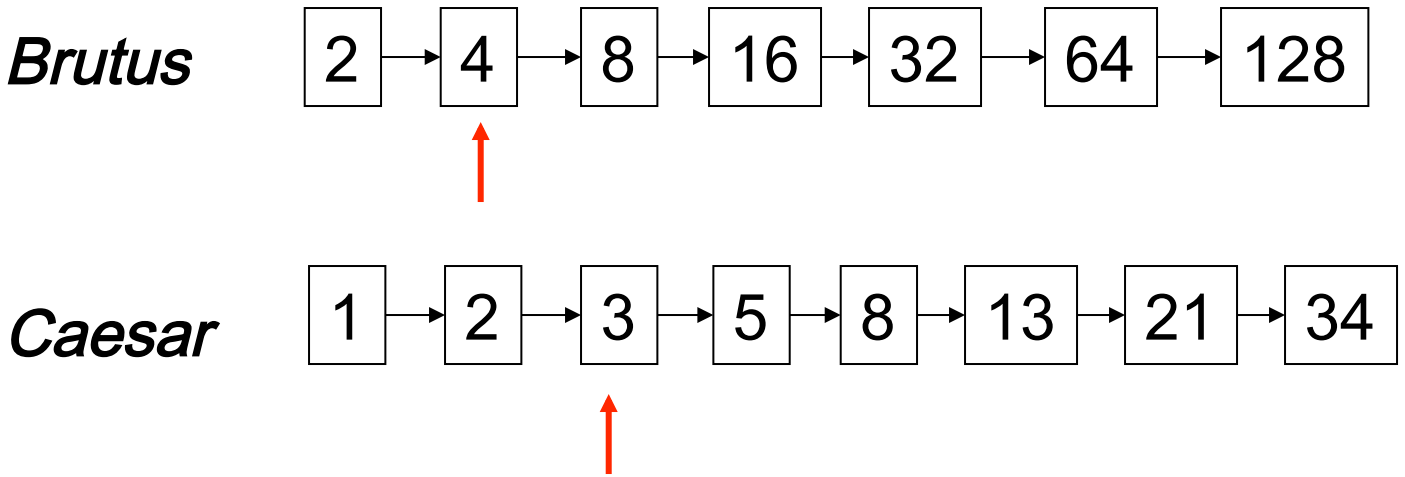
- Walk through the two postings simultaneously



Brutus AND Caesar 2

The merge

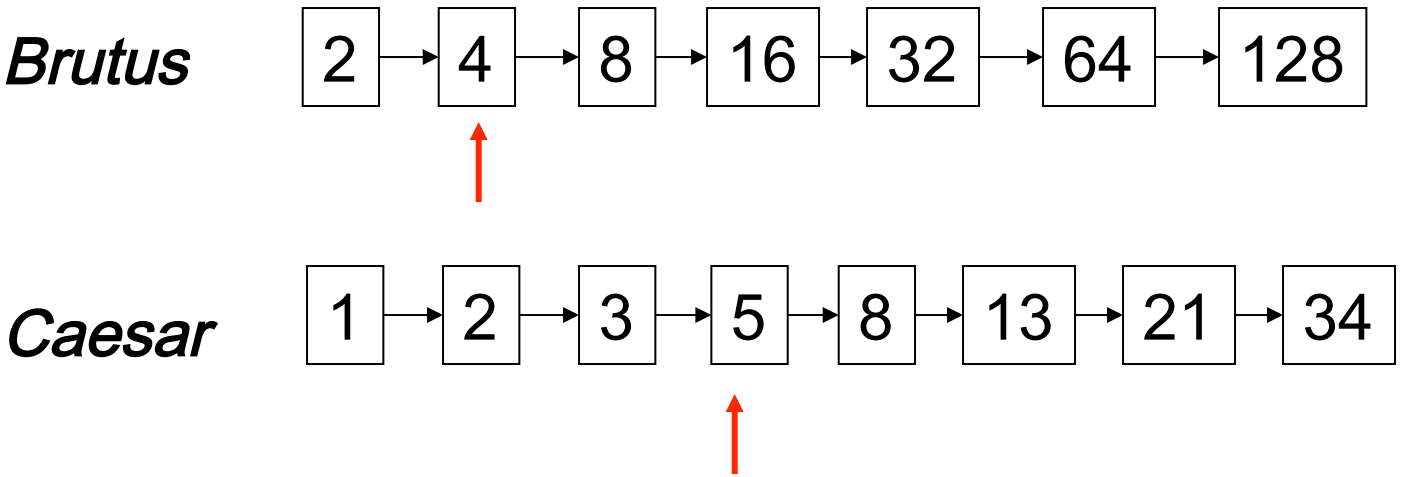
- Walk through the two postings simultaneously



Brutus AND Caesar → 2

The merge

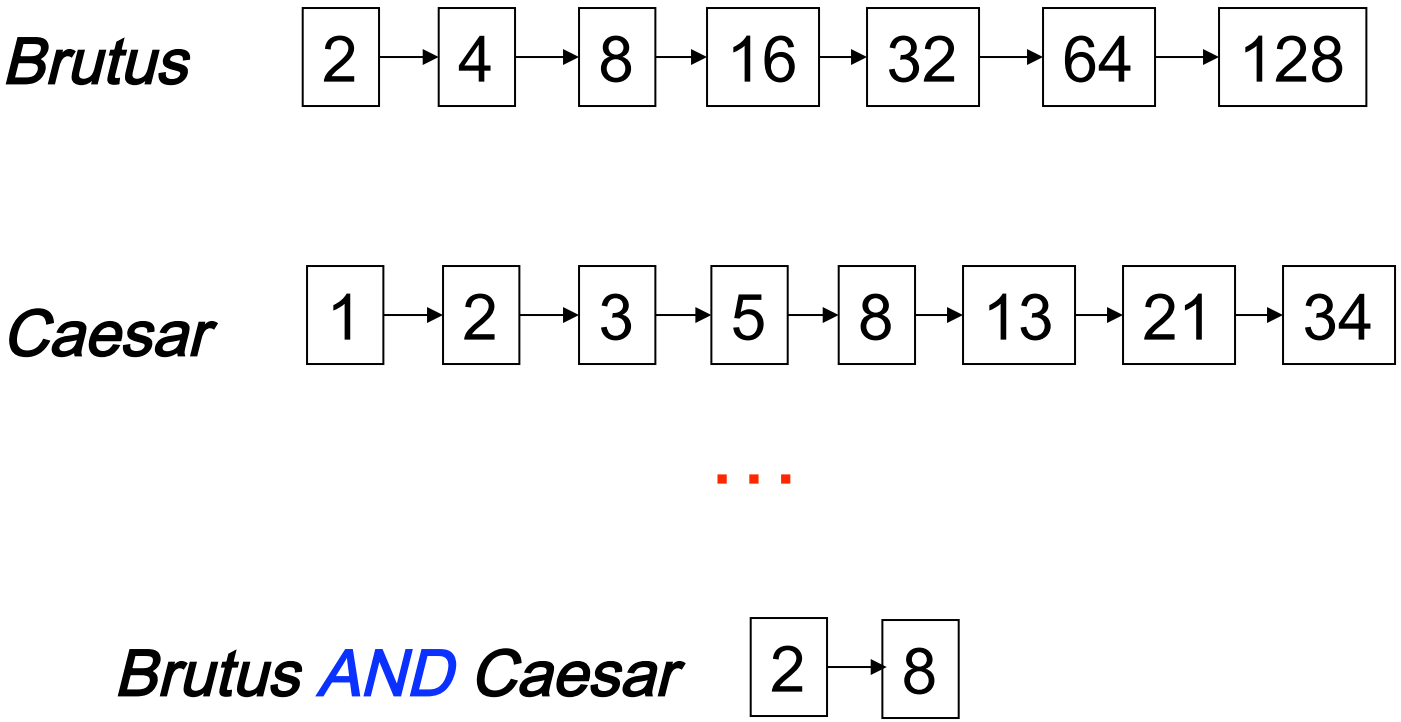
- Walk through the two postings simultaneously



Brutus AND Caesar → 2

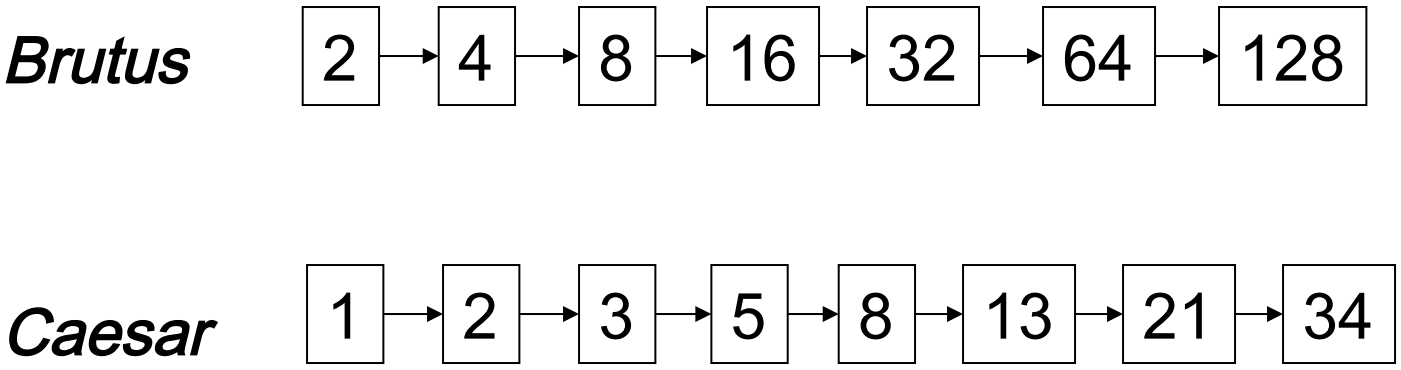
The merge

- Walk through the two postings simultaneously



The merge

- Walk through the two postings simultaneously

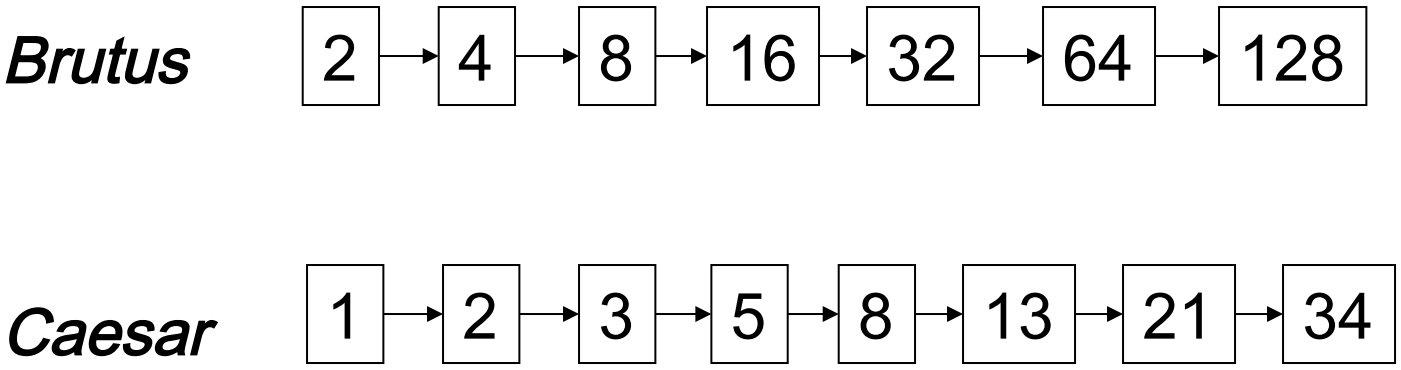


What assumption are we making about the postings lists?

For efficiency, when we construct the index, we ensure that the postings lists are sorted

The merge

- Walk through the two postings simultaneously



What is the running time?

O(length1 + length2)

Boolean queries: More general merges

- Which of the following queries can we still do in time O(length1+length2)?

Brutus *AND NOT* ***Caesar***

Brutus *OR NOT* ***Caesar***

Merging

What about an arbitrary Boolean formula?

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

(Antony** OR **Cleopatra)

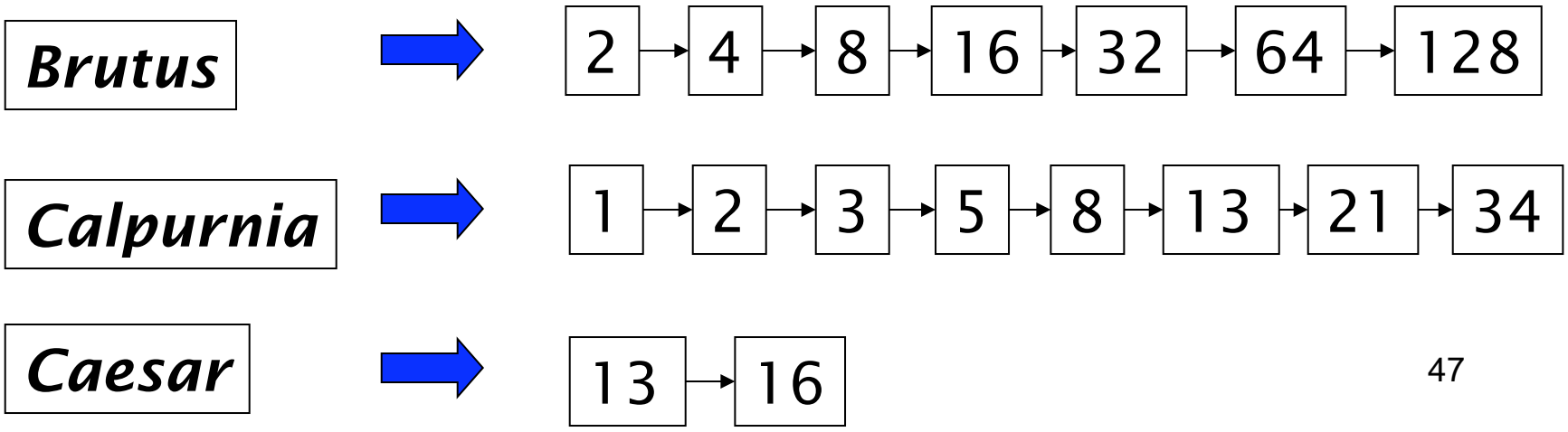
- *x* = ***(Brutus** OR **Caesar)***
- *y* = ***(Antony** OR **Cleopatra)***
- *x AND NOT y*

- Is there an upper bound on the running time?
  - O(total_terms * query_terms)
- What about ***Brutus** AND **Calpurnia** AND **Caesar***?

Query optimization

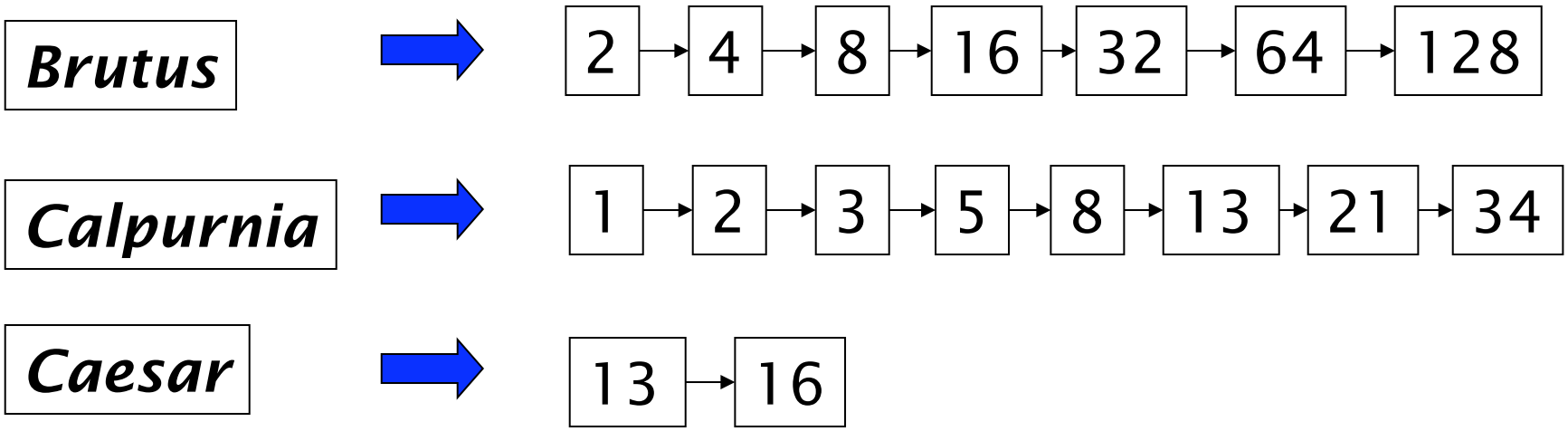
Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

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



Query optimization example

- Heuristic: Process in order of increasing freq:
  - *merge the two terms with the shortest postings list*
  - *this creates a new AND query with one less term*
  - repeat

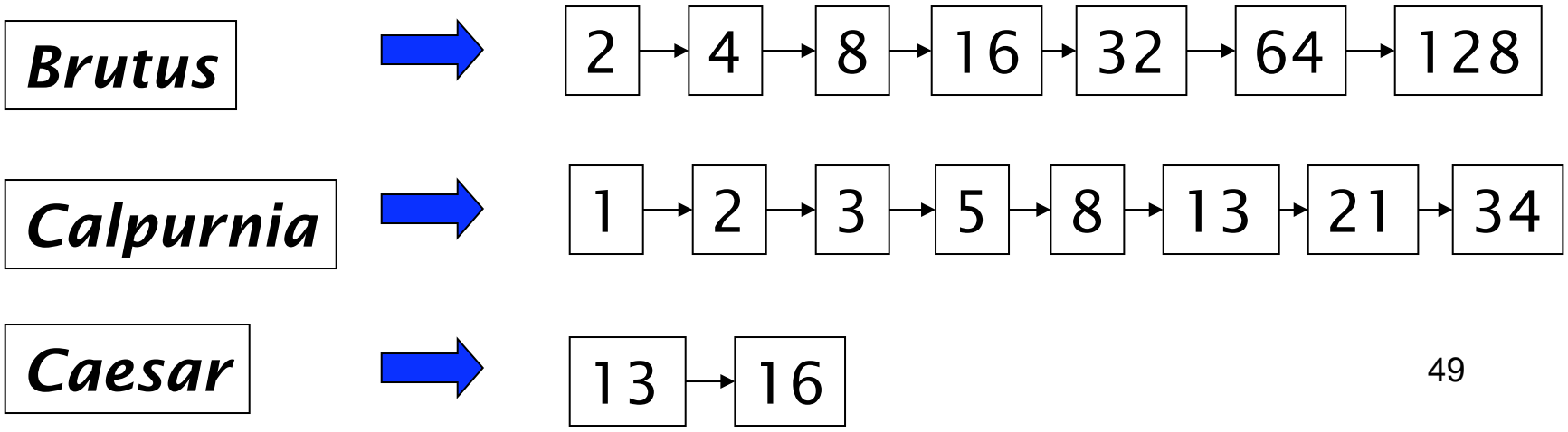


Execute the query as (***Caesar*** *AND* ***Brutus)*** *AND* ***Calpurnia***.

Query optimization

Query: ***Brutus*** *OR* ***Calpurnia*** *OR* ***Caesar***

- Consider a query that is an *OR* of *t* terms.
- What is the best order for query processing?
- Same: still want to merge the shortest postings lists first



Query optimization in general

- (madding OR crowd) AND (ignoble OR NOT strife)

- Need to evaluate OR statements first
- Which OR should we do first?
  - Estimate the size of each OR by the sum of the posting list lengths
  - NOT is just the number of documents minus the length
  - Then, it looks like an AND query:
    - *x* AND *y*

Exercise

- Recommend a query processing order for

***(tangerine** OR NOT **trees)** AND*
***(marmalade** OR **skies)** AND*
(kaleidoscope** OR **eyes)

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

Next steps…

- Phrases
  - ***Pomona College***
- Proximity: Find ***Gates*** *NEAR* ***Microsoft***.
  - Need index to capture position information in docs. More later
- Zones in documents: Find documents with (*author =* ***Ullman***) *AND* (text contains ***automata***)
- Ranking search results
  - include occurrence frequency
  - weight different zones/features differently (e.g. title, header, link text, …)
- Incorporate link structure

Resources for today's lecture

- Introduction to Information Retrieval, ch. 1
- Managing Gigabytes, Chapter 3.2
- Modern Information Retrieval, Chapter 8.2
- Shakespeare: http://www.rhymezone.com/shakespeare/
  - Try the neat browse by keyword sequence feature!