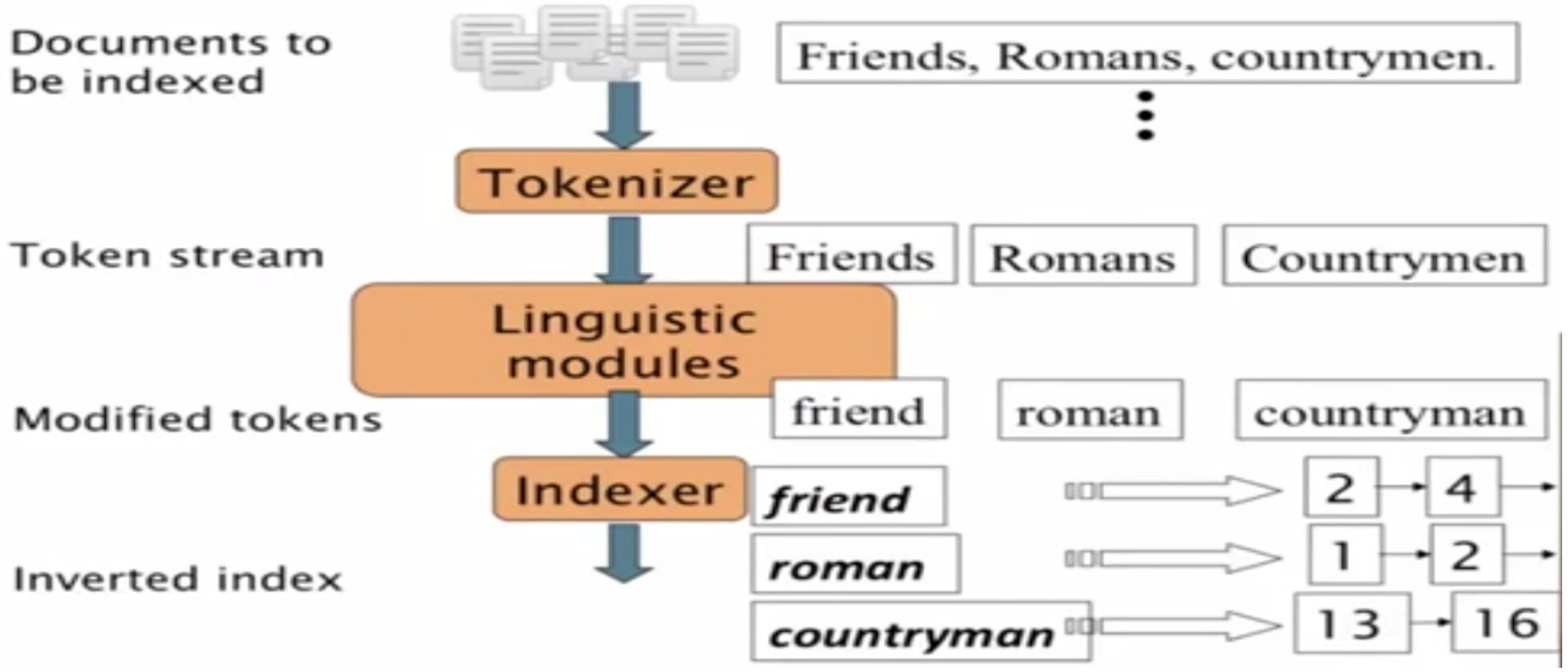
Information Retrieval

Prof: Ehab Ezzat Hassanein

Constructing And Querying Inverted Indexes

Inverted Index construction



Initial stages of text processing

- **Tokenization**
  - Cut character sequence into words tokens
    - Deal with “John’s”, a state-of-the-art solution
- **Normalization**
  - Map text and query term to the same form
    - USA and U.S.A to match
- **Stemming**
  - We may wish different forms of a root to match
    - authorize and authorization
- **Stop words**
  - We may omit very common words (or not!)
    - The, a, to, of
      - Query the song to be or not to be!!

Indexer Steps: Token Sequence

The Sequence of (modified tokens, document ID) pairs

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	docID
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

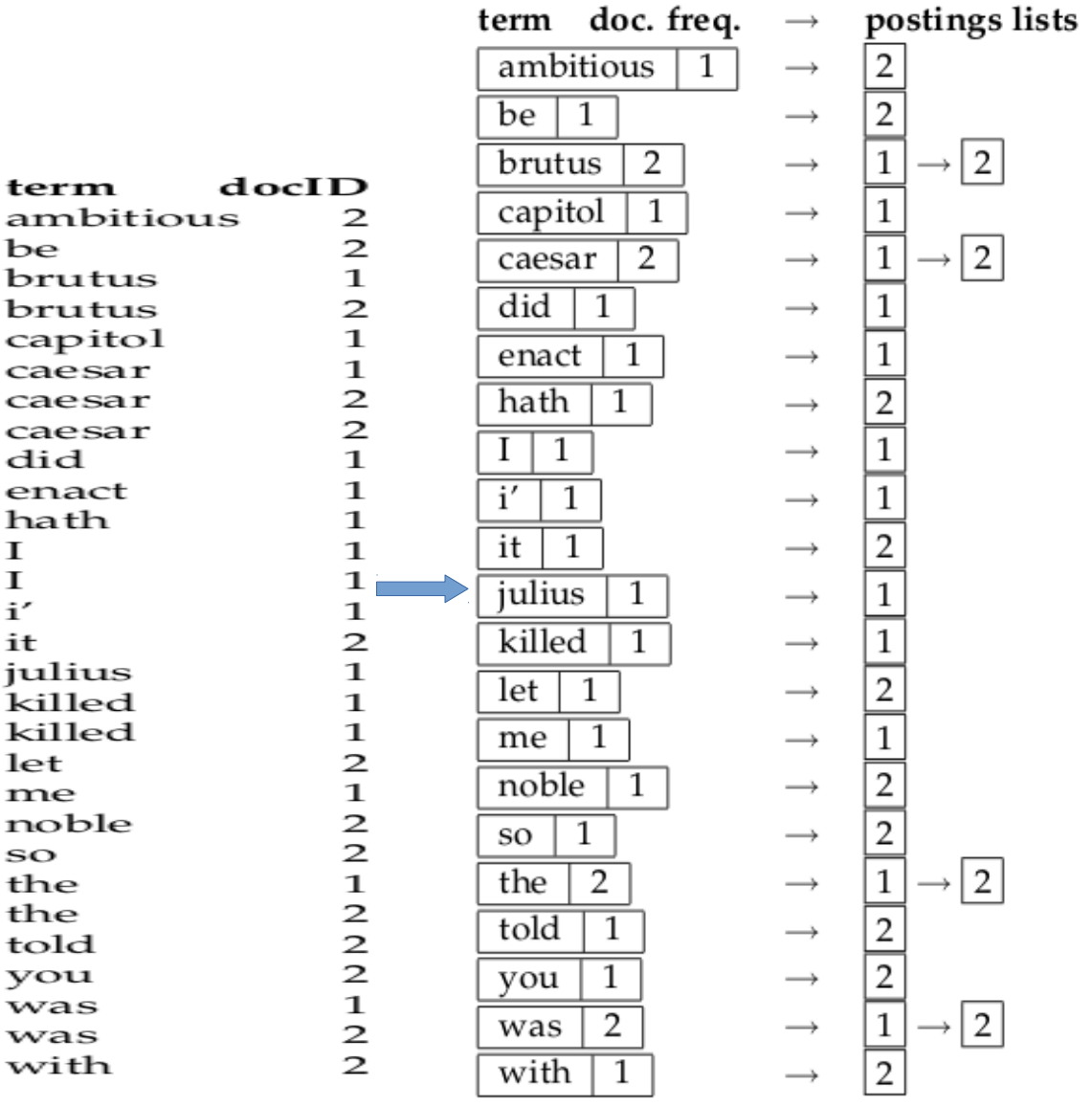
Indexer Steps: Sort

- Sort by terms
  - And then docID

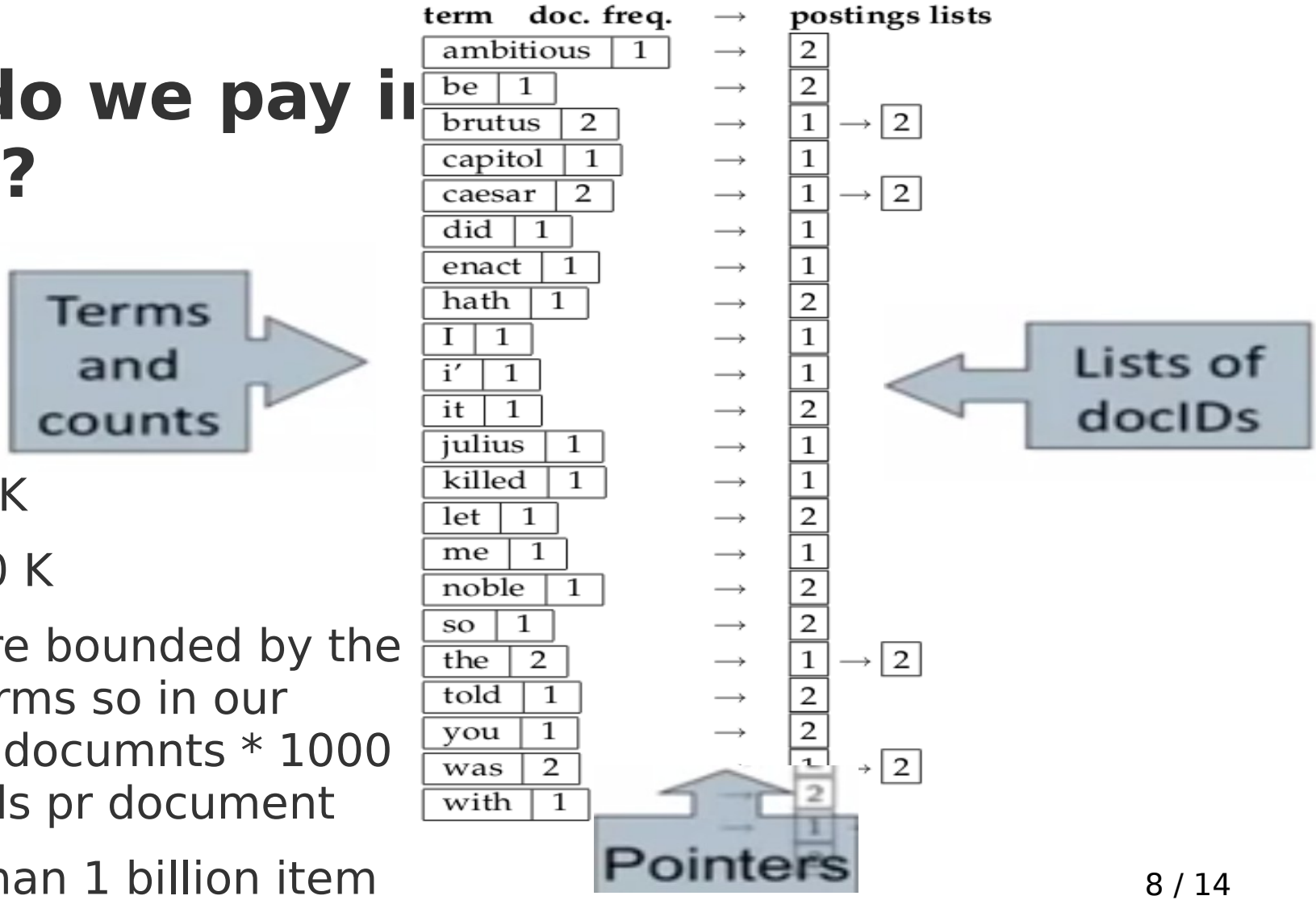
term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexer Steps: Dictionary And Postings

- Multiple term entries in a single document are merged
- Split into Dictionary and Postings
- Doc Frequency information is added



Where do we pay i
Storage?



- Terms ~ 500 K
- Pointer ~ 500 K
- Posting list are bounded by the number of terms so in our example 1M documnts * 1000 average words pr document

  ==>> less than 1 billion item

Efficient IR System Implementation

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

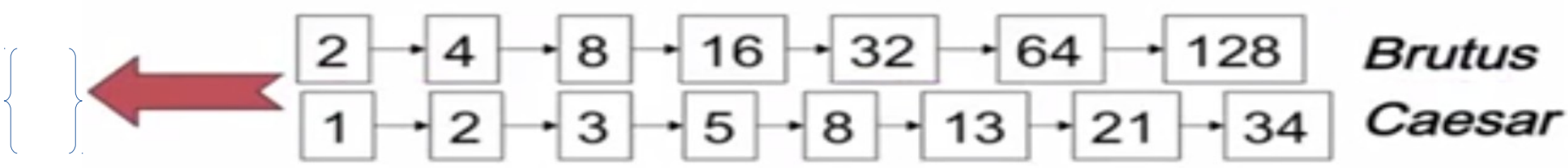
Query Processing with an Inverted Index

Query Processing: AND

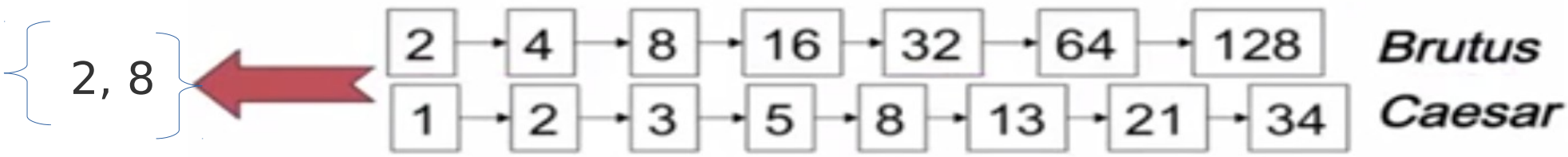
- Consider Processing query:

  Brutus and Caesar

- 1. Locate Brutus in the Dictionary
- 2. Retrieve its postings
- 3. Locate Caesar in the Dictionary
- 4. Retrieve its postings
- 5. Merge the two postings lists (intersect the document sets):



Algorithm for the merging of two postings lists



Algorithm for the merging of two postings lists

```
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
```