Information Retrieval

Abhishek
January 2020

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

CS F469, Information Retrieval
Lecture No. 4

Recap of Lecture 3

- Character encodings
- Document unit
- Tokenization
- Normalization
- Stemming and Lemmatization

Today's Lecture

- Faster merging of posting lists
- Positional postings and phrase queries
- Collocations

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1 →

Doc 1	Doc 20	Doc 34	Doc 90	**...**

intersection

Word 3 →

Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

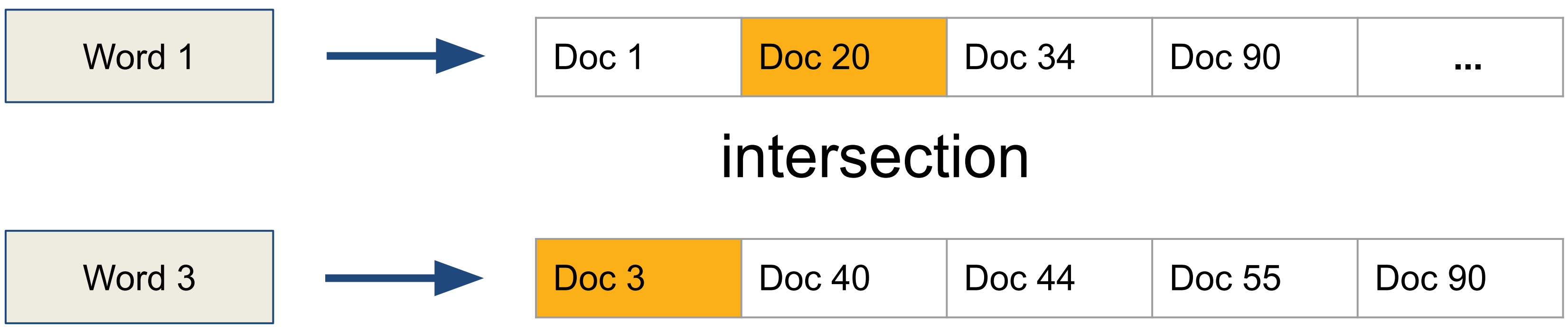
Word 1	→	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3	→	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**



Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1	→	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3	→	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1	→	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3	→	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1 →	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3 →	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1	→	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3	→	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1 →	Doc 1	Doc 20	Doc 34	**Doc 90**	...

intersection

Word 3 →	Doc 3	Doc 40	Doc 44	**Doc 55**	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1 →	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3 →	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1 → | Doc 1 | Doc 20 | Doc 34 | Doc 90 | ... |

intersection

Word 3 → | Doc 3 | Doc 40 | Doc 44 | Doc 55 | Doc 90 | NIL |

Result → | Doc 90 |

Recap: Posting List Intersection

Eg. query: **word 1** AND **word 3**

Word 1 →

Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3 →

Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

=

Result → Doc 90

The algorithm will perform $O(|L_1| + |L_2|)$ operations.

Skip Pointers in Posting Lists

Faster Posting List Intersection via Skip Pointers

Brutus →

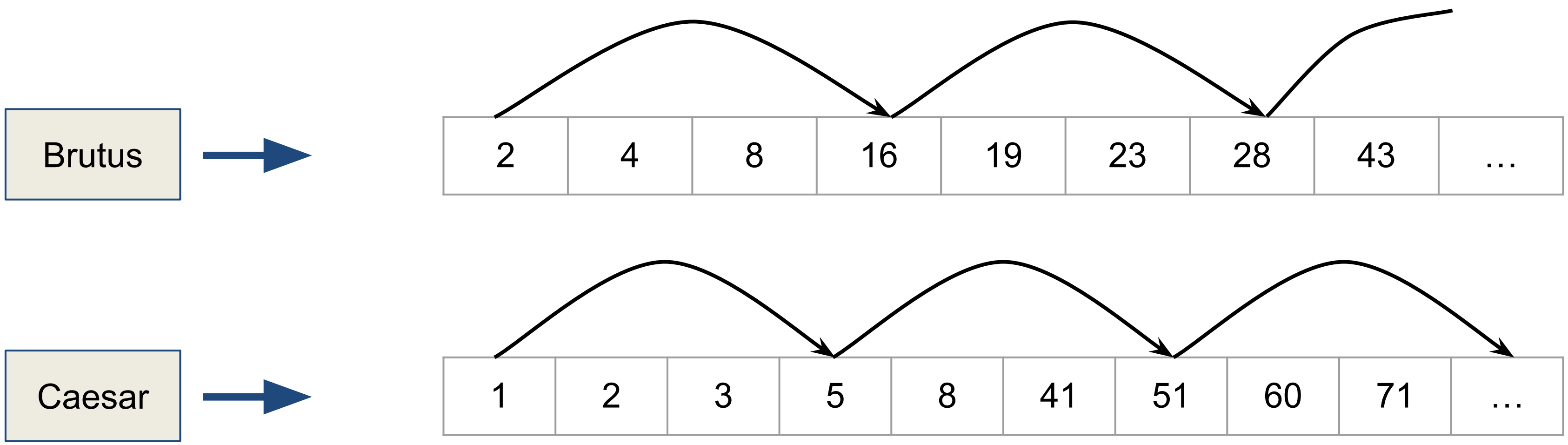
2	4	8	16	19	23	28	43	…

Caesar →

1	2	3	5	8	41	51	60	71	…

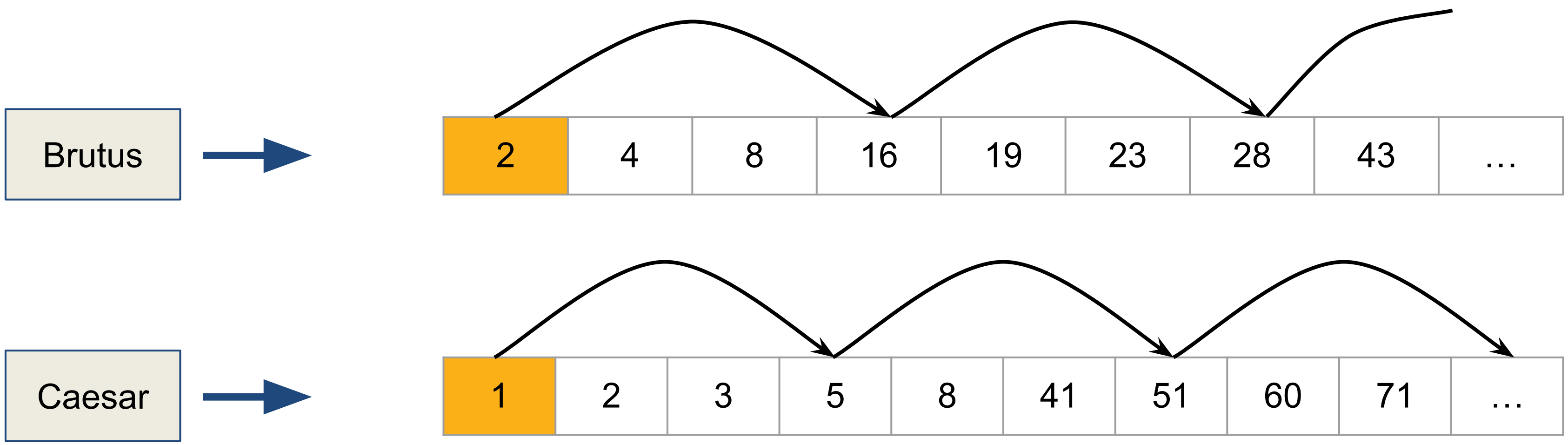
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



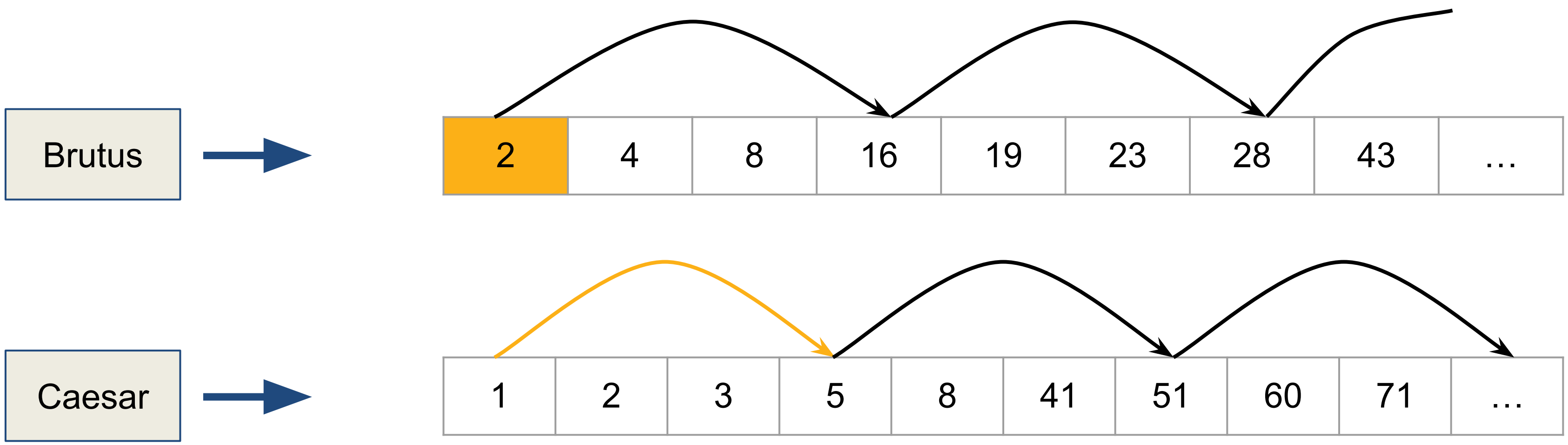
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



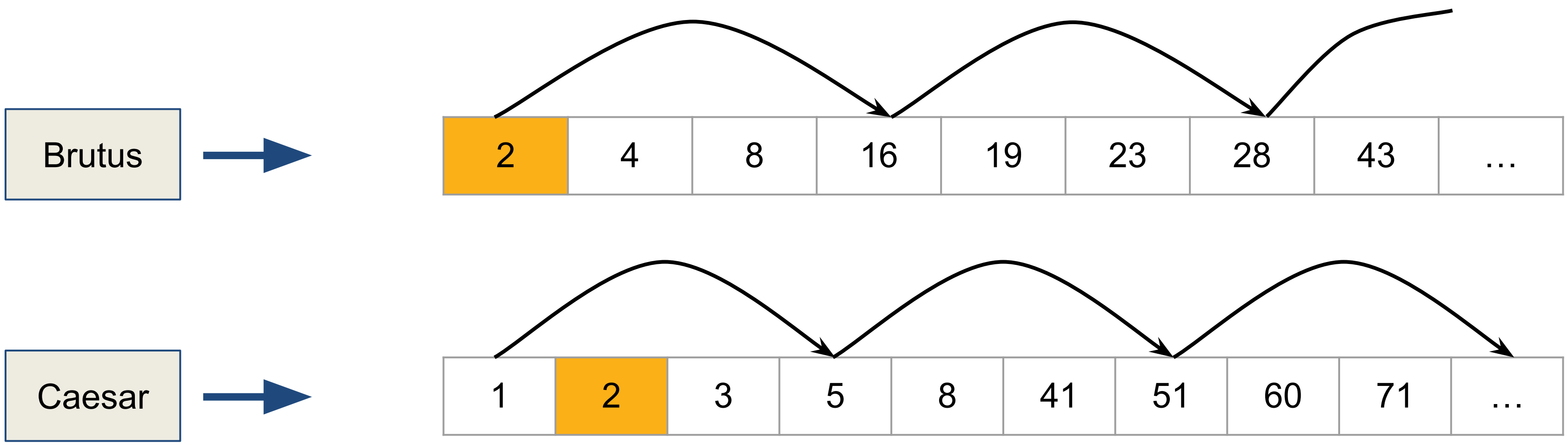
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



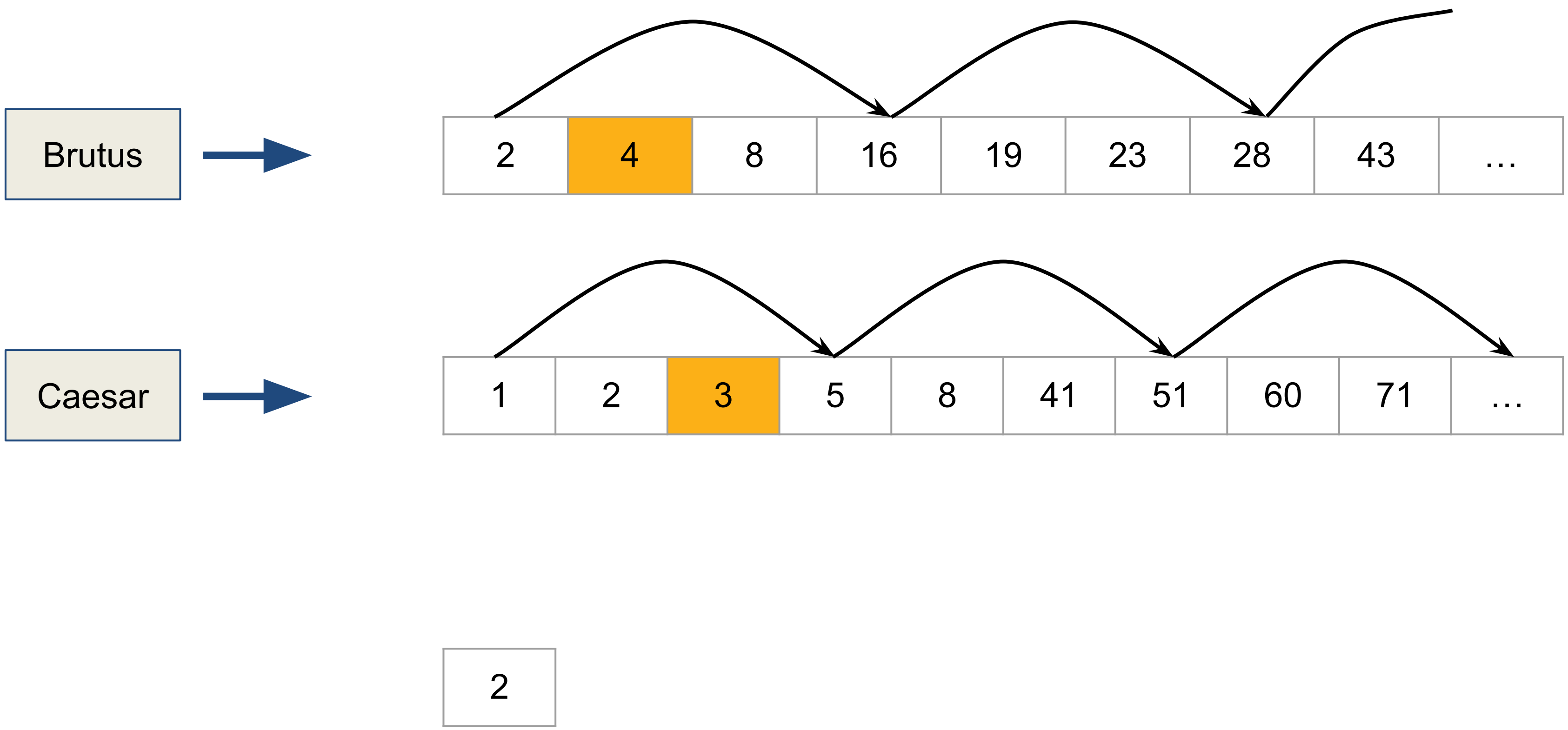
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



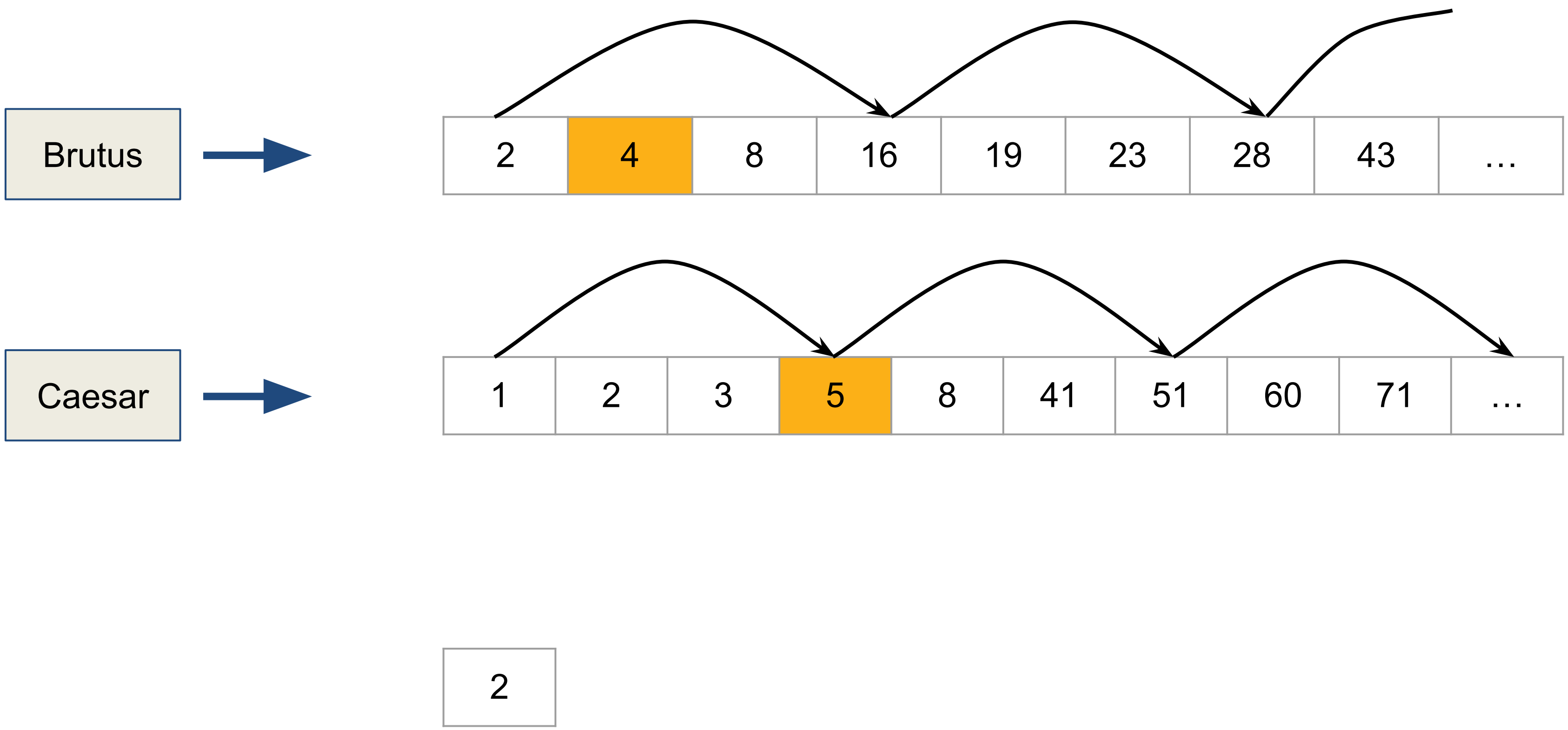
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



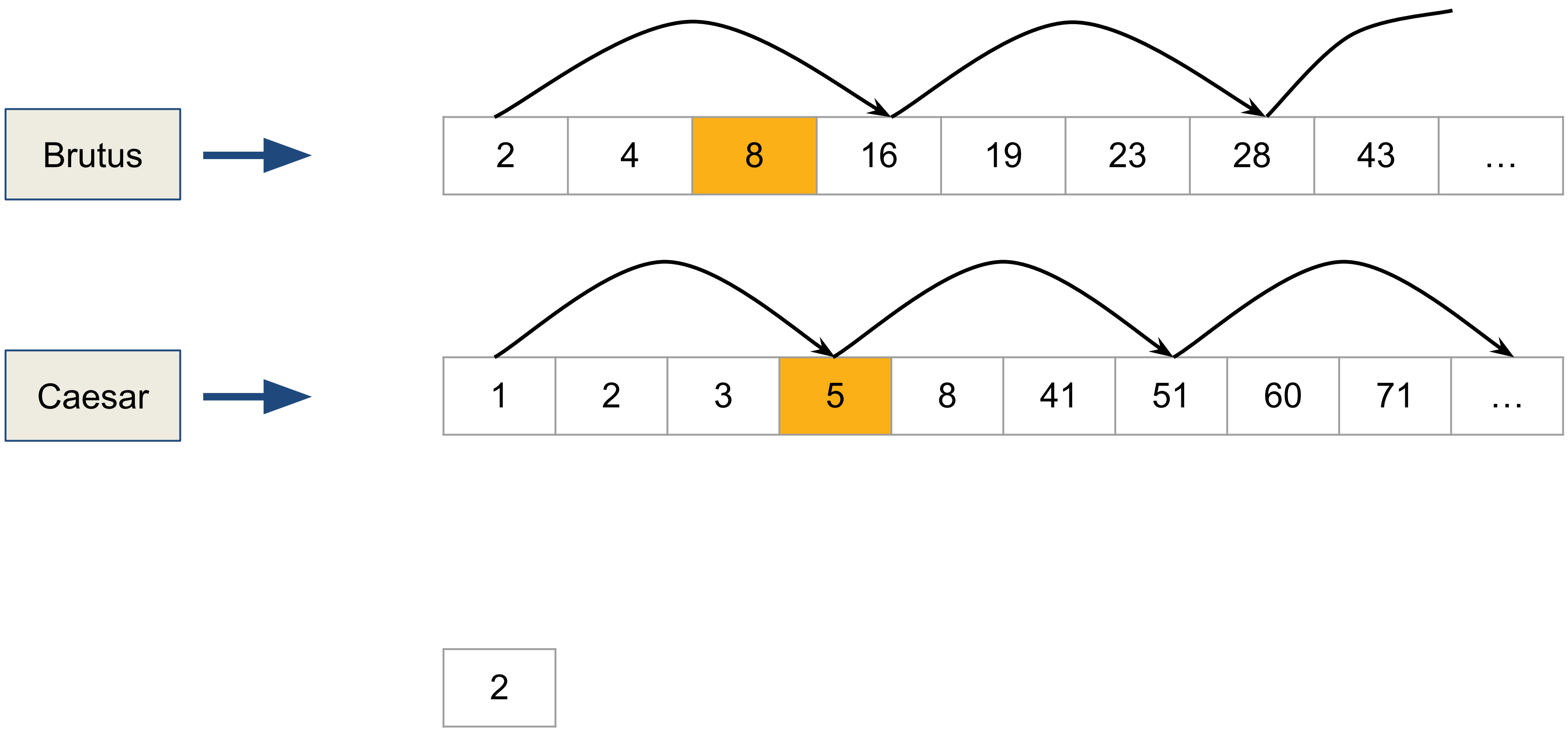
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



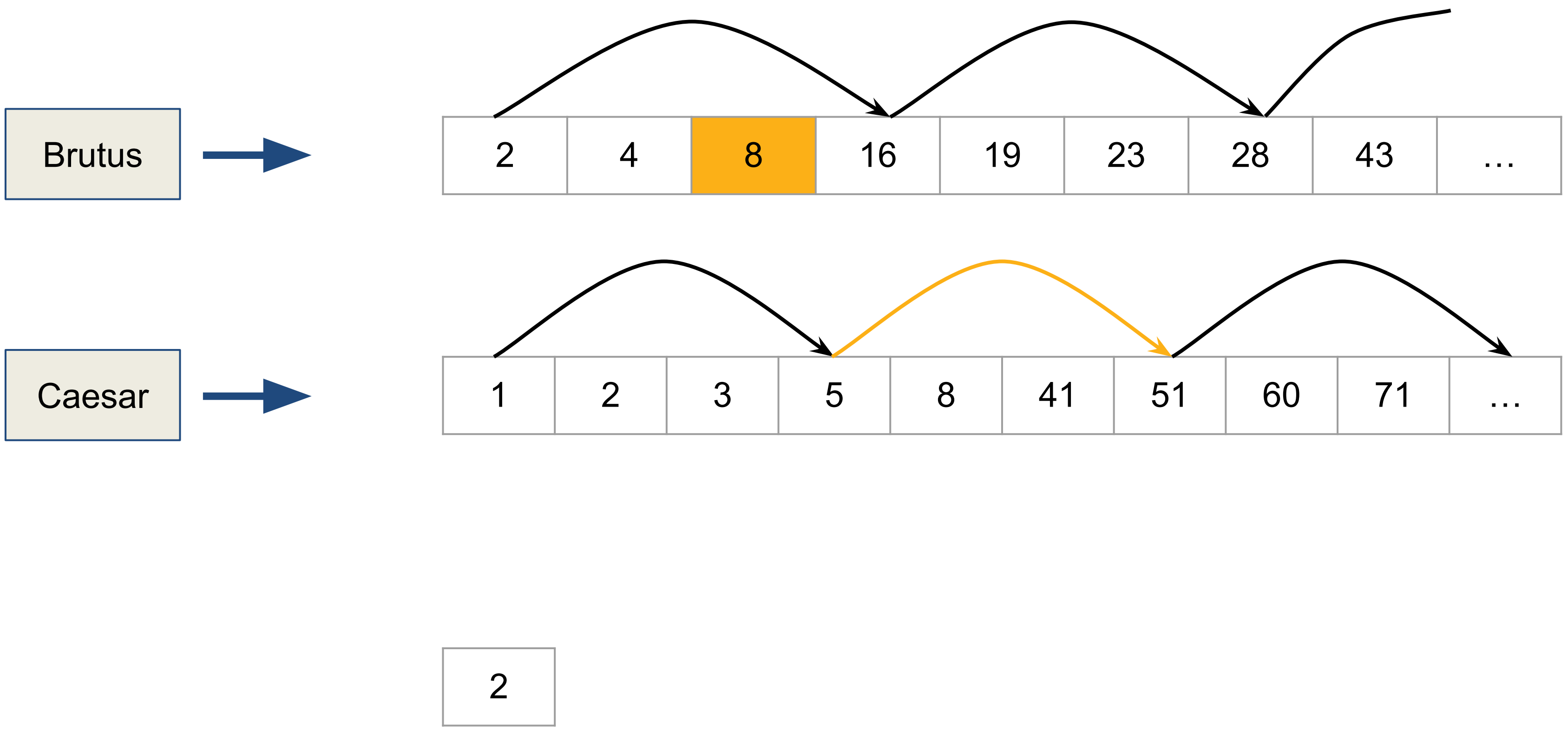
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



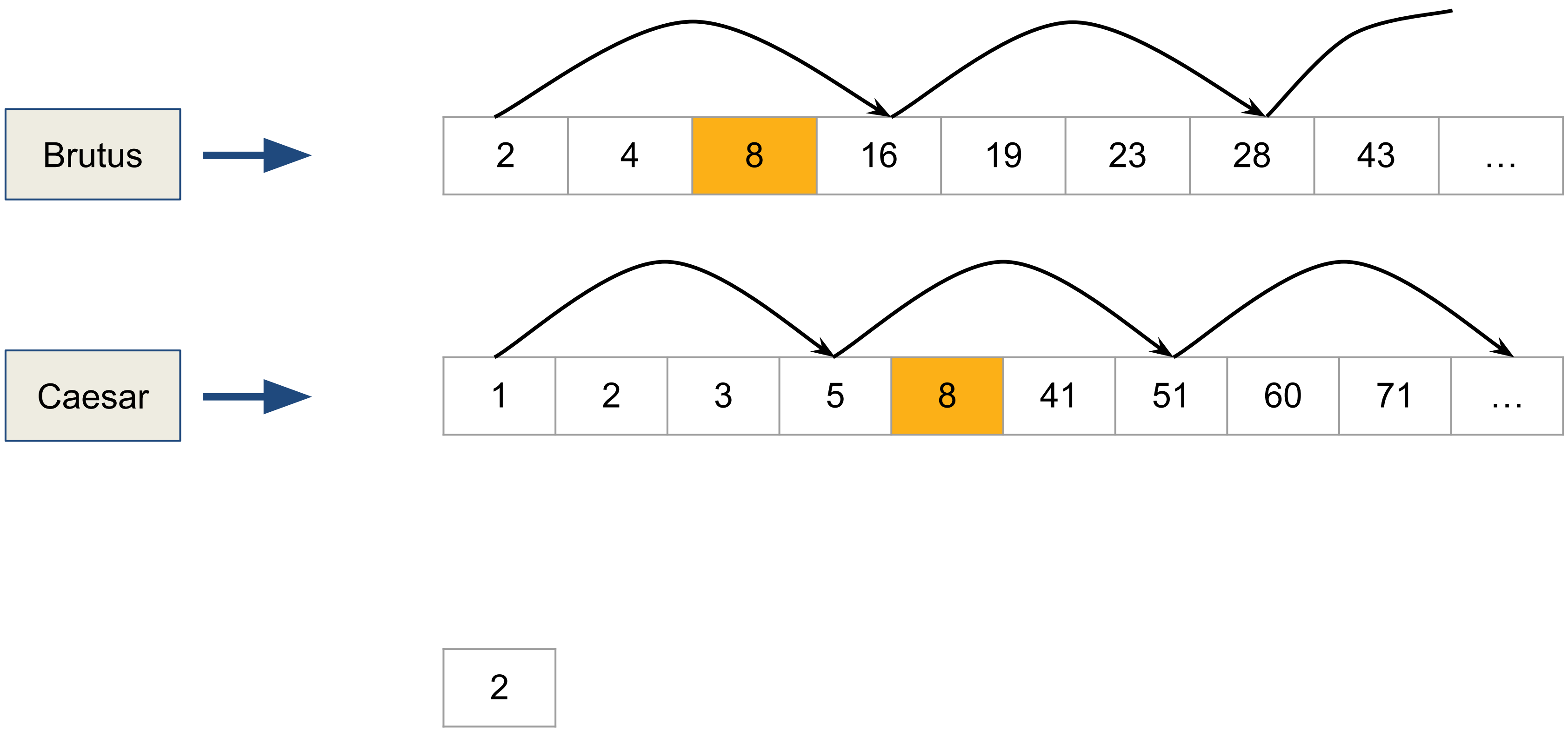
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



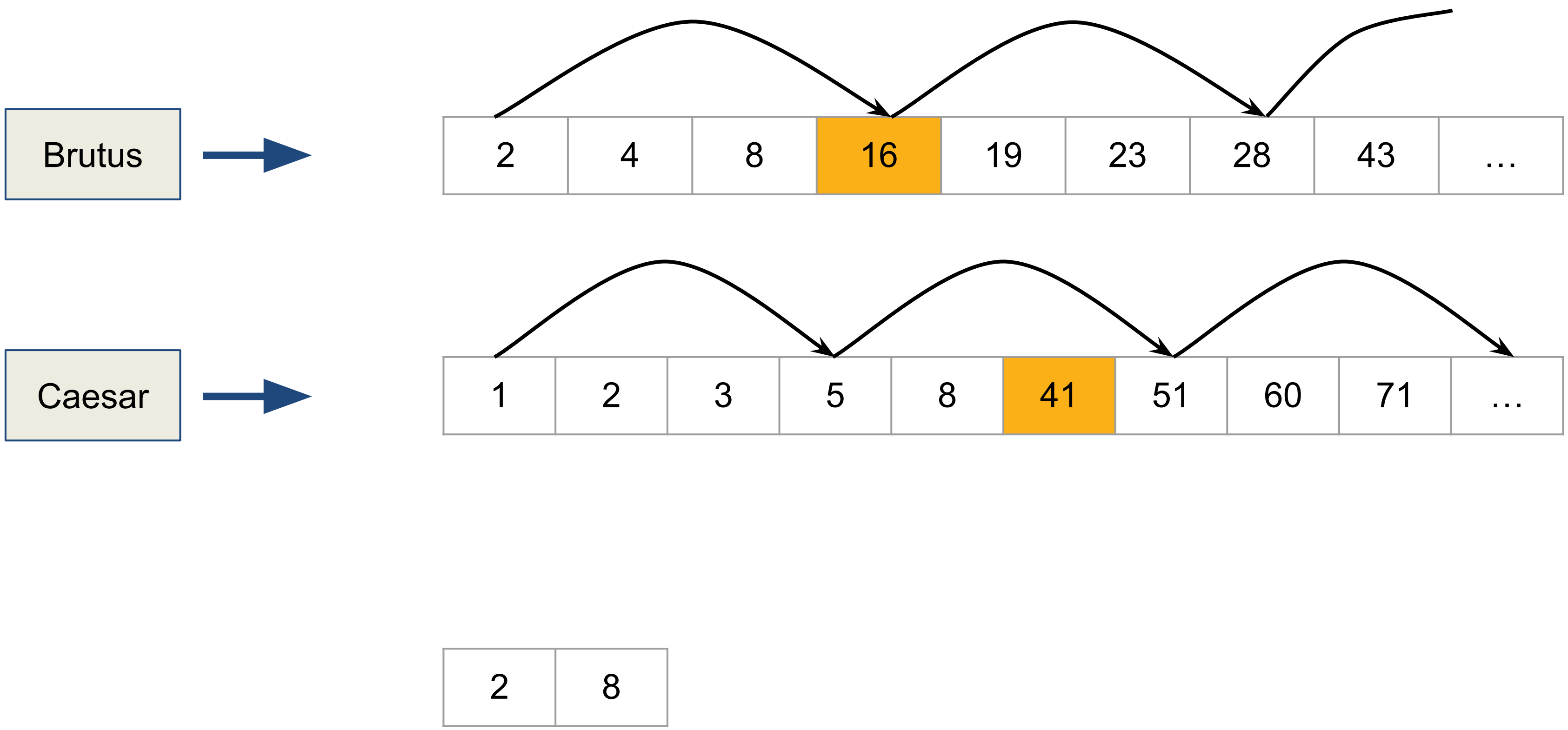
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



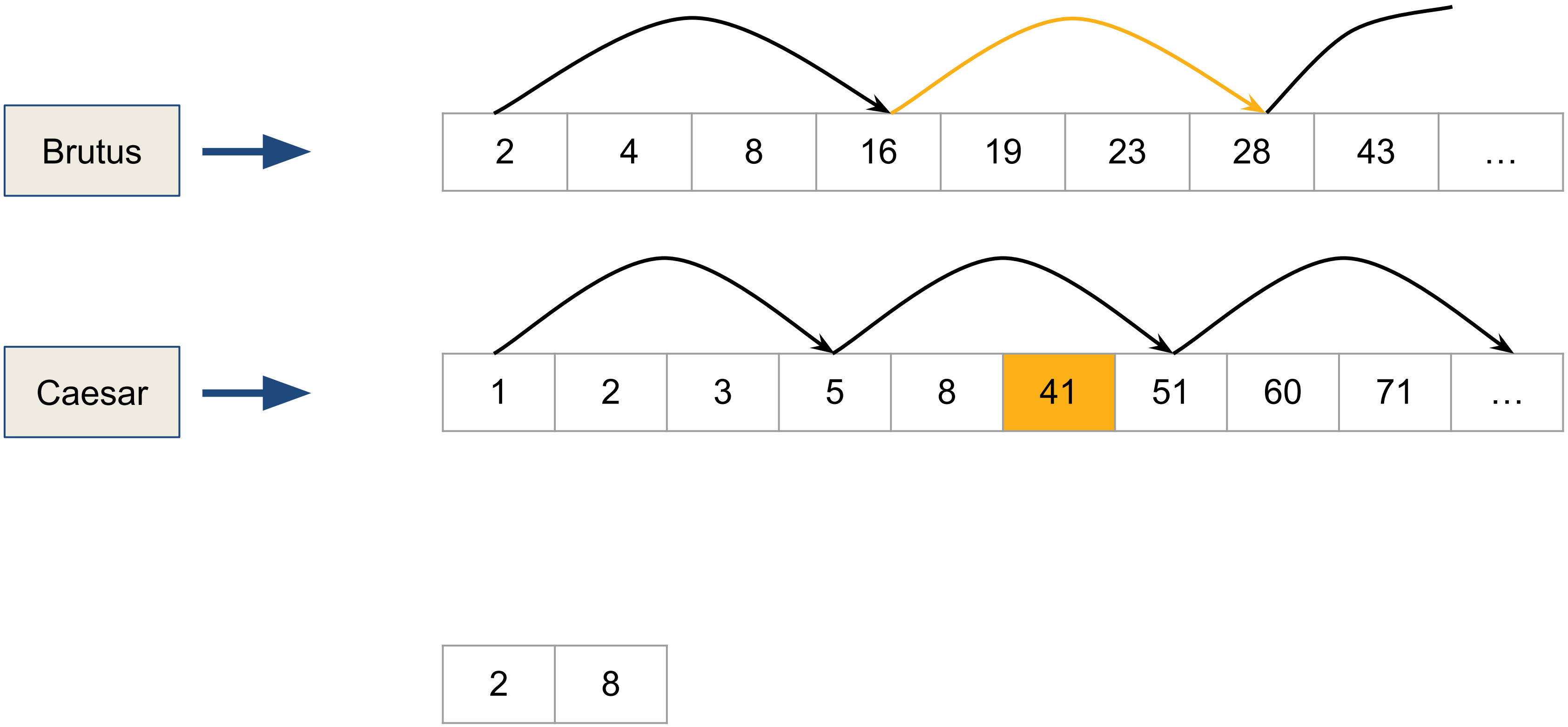
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



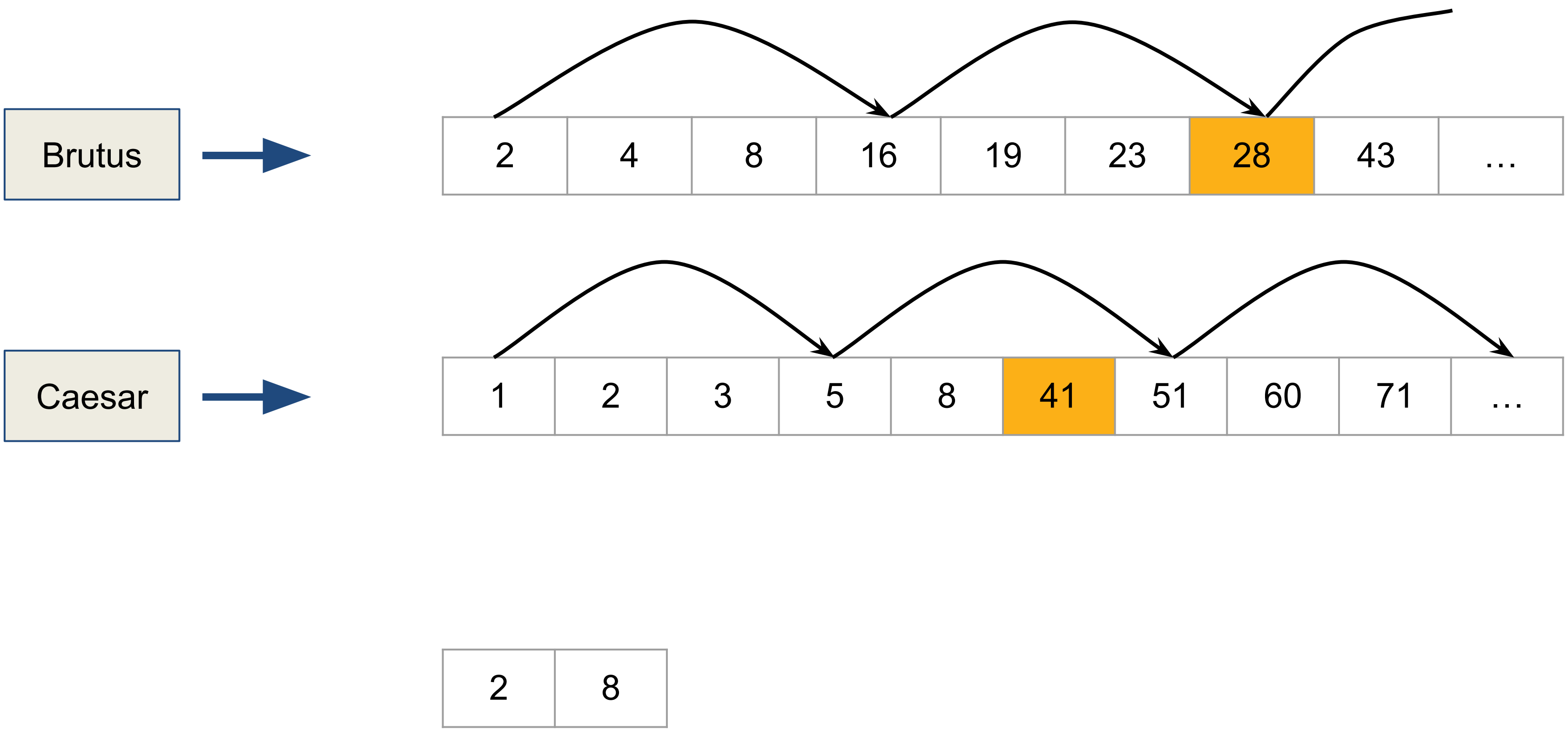
Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Faster Posting List Intersection via Skip Pointers



Example from Figure 2.9, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Algorithm

```
INTERSECTWITHSKIPS(p1, p2)
answer ← ⟨ ⟩
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
     then ADD(answer, docID(p1))
          p1 ← next(p1)
          p2 ← next(p2)
     else if docID(p1) < docID(p2)
            then if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
                   then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
                          do p1 ← skip(p1)
                   else p1 ← next(p1)
            else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
                   then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
                          do p2 ← skip(p2)
                   else p2 ← next(p2)
return answer
```

Figure 2.10, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Where to place skip pointers?

- Tradeoff
  - **Too many skip pointers** → Shorter skip span

    Skip pointers will be used more and also number of comparisons related to skip pointers will be high.

  - **Too few skip pointers** → Longer skip span

    Few successful long skips, however most cases will be like linear scan.

Where to place skip pointers?

- Tradeoff
  - **Too many skip pointers** → Shorter skip span

    Skip pointers will be used more and also number of comparisons related to skip pointers will be high.

  - **Too few skip pointers** → Longer skip span

    Few successful long skips, however most cases will be like linear scan.

- A simple heuristics that work well in practice: For posting list of size P, use sqrt(P) evenly spaced skip pointers.[1]

[1]Moffat, Alistair, and Justin Zobel. "Self-indexing inverted files for fast text retrieval." *ACM Transactions on Information Systems (TOIS)* 14.4 (1996): 349-379.

Limitations

- Harder to maintain skip pointers if the index is changing frequently because of updates in the documents.
- The posting list size is larger, as these additional pointers need to be stored.
- In modern systems, the I/O cost of loading bigger posting list can be much larger than the benefits gained for quicker in-memory merges.

Questions

- Will skip pointers be useful for queries of the form **x** OR **y**?

- Does skip pointer-based intersection algorithm uses an order of magnitude fewer operations on its worst-case when compared with the normal merge algorithm worst-case?

Positional Postings and Phrase Queries

Phrase Queries

- Several group of words are phrases, i.e., group of words standing together as a conceptual unit. For example, **Stanford University**, **New Delhi**, **New York**.
- We would like that the IR systems, do not consider them as words connected by AND operator.

Phrase Queries

- Several group of words are phrases, i.e., group of words standing together as a conceptual unit. For example, **Stanford University**, **New Delhi**, **New York**.
- We would like that the IR systems, do not consider them as words connected by AND operator.

Example Query: "Stanford University"
Example Document: "The inventor Stanford Ovshinsky never went to university."

- Shall we consider the above document relevant for the example query?

- **Biword indexes**
- **Positional postings**

Biword indexes

- Consider every pair of consecutive terms in a document as a phrase.

For example, the text Friends, Romans, Countrymen would generate the biwords:

friends romans
romans countrymen

- Two word phrasal queries can be easily answered.

Processing longer phrases in Biword indexes

- Longer phrases can be processed by breaking them down.
- For example: “stanford university palo alto” phrase can be decomposed as:

“stanford university” AND “university palo” AND “palo alto”

- Does the documents retrieved by the above decomposed query always contains the original phrase?

Processing longer phrases in Biword indexes

- Longer phrases can be processed by breaking them down.
- For example: “stanford university palo alto” phrase can be decomposed as:

“stanford university” AND “university palo” AND “palo alto”

- Does the documents retrieved by the above decomposed query always contains the original phrase? **NO**

Extended Biword index

- Phrases of form: “the abolition of slavery”, “renegotiation of the constitution”.
- Here nouns and noun phrases which describe important concepts are separated from each other by various function words.
- In such cases, use Part-Of-Speech tagging.

renegotiation	of	the	constitution
N	X	X	N

- Now any string of term form NX*N can be considered as a biword.

Extended Biword index

Example:

The query,

cost overruns on a power plant

is parsed into

“cost overruns” AND “overruns power” AND “power plant”

Limitations of Biword Indexes

- How can we retrieve documents for single term query using biword indexes?
- Index size blow up due to large vocabulary of biwords.
- Is effective for phrases of more that two words, however, does not always return correct answers.

Positional postings

Positional Indexes

- While indexing a term, also store the position where the term appeared in the document.

Positional Indexes

to, 993427:

(1, 6: (7, 18, 33, 72, 86, 231);

2, 5: (1, 17, 74, 222, 255);

4, 5: (8, 16, 190, 429, 433);

5, 2: (363, 367);

7, 3: (13, 23, 191); …)

be, 178239:

(1, 2: (17, 25);

4, 5: (17, 191, 291, 430, 434);

5, 3: (14, 19, 101); …)

Example from Figure 2.11, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Positional Indexes

to, 993427:

(1, 6: (7, 18, 33, 72, 86, 231);
2, 5: (1, 17, 74, 222, 255);
4, 5: (8, 16, 190, 429, 433);
5, 2: (363, 367);
7, 3: (13, 23, 191); …)

be, 178239:

(1, 2: (17, 25);
4, 5: (17, 191, 291, 430, 434);
5, 3: (14, 19, 101); …)

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

Example from Figure 2.11, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Positional Indexes

to, 993427:

(1, 6: (7, 18, 33, 72, 86, 231);

2, 5: (1, 17, 74, 222, 255);

4, 5: (8, 16, 190, **429**, **433**);

5, 2: (363, 367);

7, 3: (13, 23, 191); …)

be, 178239:

(1, 2: (17, 25);

4, 5: (17, 191, 291, **430**, **434**);

5, 3: (14, 19, 101); …)

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

Example from Figure 2.11, Introduction to IR, C.D. Manning, P. Raghavan and H. Schütze.

Proximity Search

- Positional indexes can be used for proximity searches.
- Example: house /4 brick
  Find all documents that contain house and brick within four words of each other.

  Relevant phrases could be:

      red brick house
      red house of brick
      house made of red brick

Combination schemes

- Usually biword indexing and positional indexing can be combined.
- There are several biwords that are extremely frequent. For example, New York, Narendra Modi, Donald Trump.
- **Combination scheme:** Index frequent biwords and other phrases using positional indexes.
- The most speedup will be when the two words are common but the desired phrase is rare. Example: “The Who”.

Collocations

- In corpus linguistics, a **collocation** is a series of words or terms that co-occur more often than would be expected by chance.

Example: Red Wine, New York, strong tea, powerful computer, heavy rail.
vs Yellow Wine, Latest York, powerful tea, strong computer, thick rain.

How to Find Collocations?

- Bigram Frequency based methods.
- Bigram frequency + POS tag frequency.

Advantages:

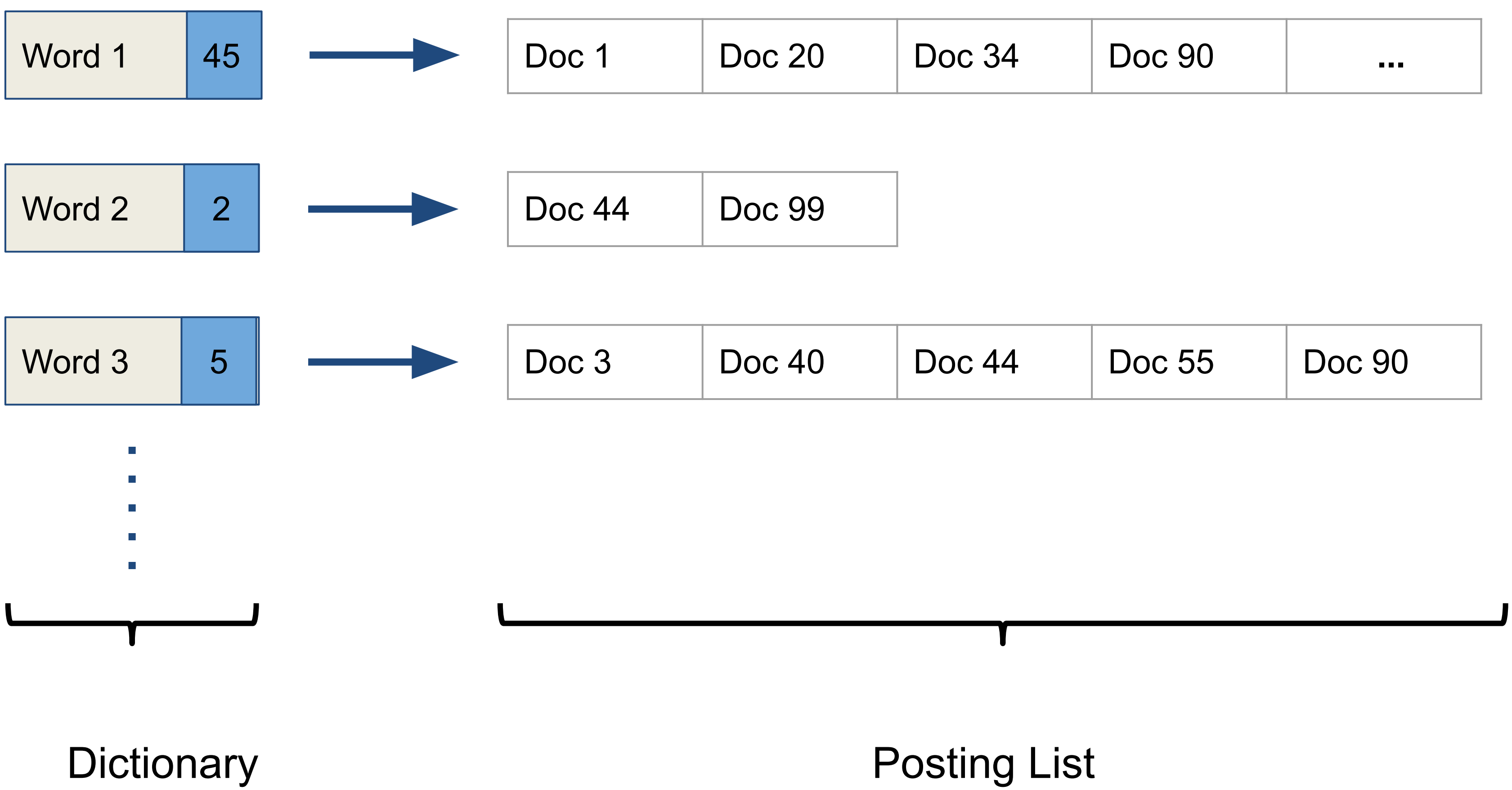
- Simple methods

Disadvantages:

- High frequency can sometimes be random, without any specific meanings.
- Works only for fixed length phrases but not for:
  - She <u>knocked</u> on his <u>door</u>.
  - They <u>knocked</u> at the <u>door</u>.

Dictionaries, Tolerant Retrieval

Dictionary data structures for inverted index



What could be underlying implementation?

- How to store a dictionary in memory efficient way?
- How do we quickly lookup elements from the dictionary?

What could be underlying implementation?

- How to store a dictionary in memory efficient way?
- How do we quickly lookup elements from the dictionary?

- Hash tables
- Trees

Reference

https://nlp.stanford.edu/IR-book/

Chapter 2

Thank You!