

Assignment Project Exam Help

Add WeChat powcoder

Introduction to

Assignment Project Exam Help

# Information Retrieval

<https://powcoder.com>

Add WeChat powcoder  
Lecture 1: Boolean retrieval

# Assignment Project Exam Help

## 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 lines containing ***Calpurnia***?  
<https://powcoder.com>
- Why is that not the answer?
  - Slow (for large corpora)
  - NOT ***Calpurnia*** is non-trivial
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Ranked retrieval (best documents to return)
    - Later lectures

# Assignment Project Exam Help

## Term-document incidence

	Antony and Cleopatra	Julius Caesar	The Taming of the Shrew	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*

Add WeChat powcoder

1 if play contains word, 0 otherwise

# Assignment Project Exam Help

## Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for ***Brutus***, ***Caesar*** and ***Calpurnia*** (complemented)  $\rightarrow$  bitwise AND.
- 110100 AND 110111 AND 101111 = 100100.

Add WeChat powcoder

# Assignment Project Exam Help

## Answers to query

- Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,

Assignment Project Exam Help  
When Antony found Julius **Caesar** dead,

He cried almost to roaring; and he wept

When at Philippi he found **Brutus** slain.

Add WeChat powcoder

- Hamlet, Act III, Scene ii

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



# Assignment Project Exam Help

## Basic assumptions of Information Retrieval

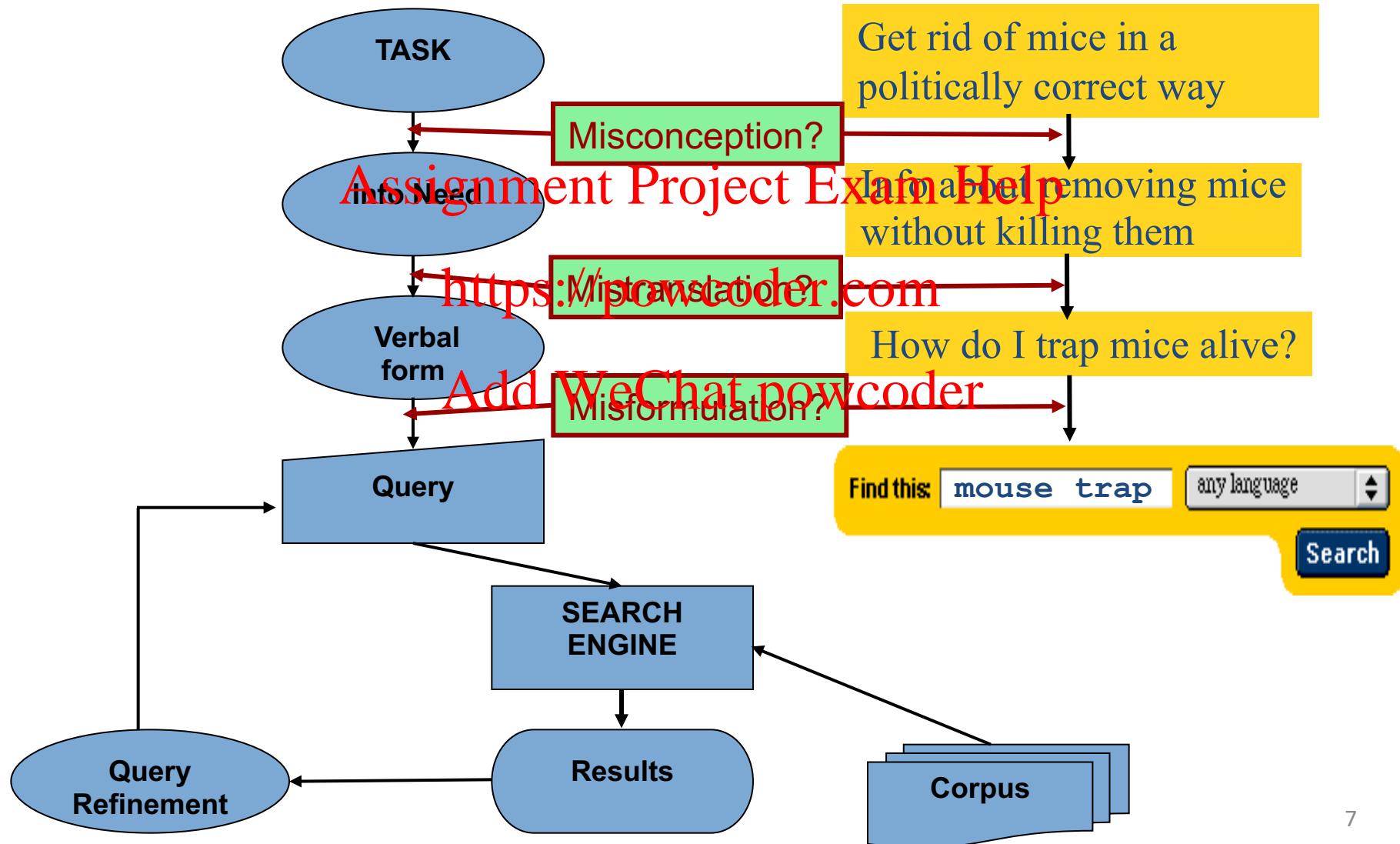
---

- **Collection:** Fixed set of documents
- **Goal:** Retrieve documents with information that is relevant to the user's **information need** and helps the user complete a task  
<https://powcoder.com>

Add WeChat powcoder

# Assignment Project Exam Help

# The classic search model



# Assignment Project Exam Help

## How good are the retrieved docs?

---

- *Precision* : Fraction of retrieved docs that are relevant to user's information need
- *Recall* : Fraction of relevant docs in collection that are retrieved <https://powcoder.com>
- More precise definitions and measurements to follow in later lectures

# Assignment Project Exam Help

## Bigger collections

- Consider  $N = 1$  million documents, each with about 1000 words.
- Avg 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents.  
<https://powcoder.com>
- Say there are  $M = 500K$  *distinct* terms among these.  
Add WeChat powcoder

# Assignment Project Exam Help

## Can't build the matrix

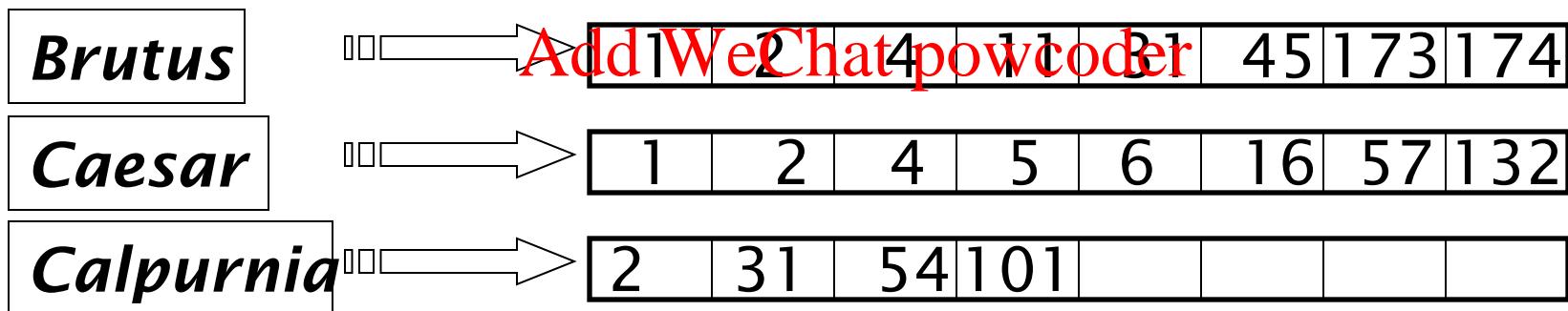
- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's.
  - matrix is extremely sparse.
- What's a better representation?
  - We only record the 1 positions.



# Assignment Project Exam Help

## Inverted index

- For each term  $t$ , we must store a list of all documents that contain  $t$ .
  - Identify each by a docID, a document serial number
- Can we use fixed-size arrays for this?  
<https://powcoder.com>



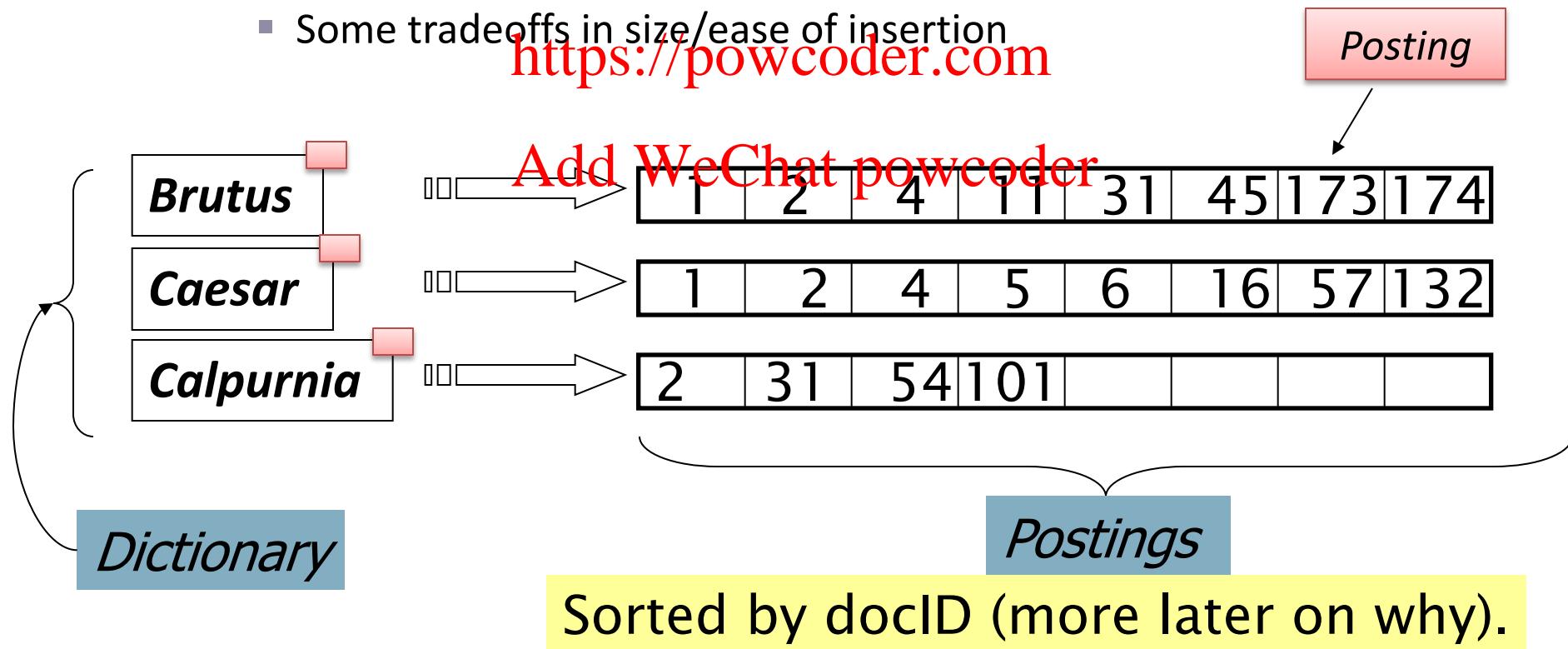
What happens if the word **Caesar** is added to document 14?

# Assignment Project Exam Help

## Inverted index

- We need variable-size postings lists
  - On disk, a continuous run of postings is normal and best
  - In memory, can use linked lists or variable length arrays
    - Some tradeoffs in size/ease of insertion

<https://powcoder.com>



# Assignment Project Exam Help

## Inverted index construction

Documents to be indexed.



Friends, Romans, countrymen.

Token stream.

Assignment Project Exam Help

Tokenizer

https://powcoder.com

Friends

Romans

Countrymen

*More on these later.*

Add WeChat powcoder  
Linguistic modules

Modified tokens.

friend

roman

countryman

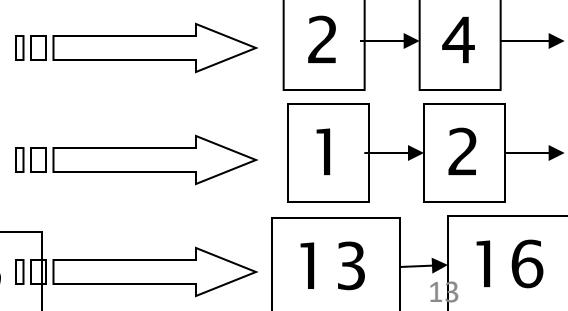
Inverted index.

Indexer

*friend*

*roman*

*countryman*



# Assignment Project Exam Help

## Indexer steps: Token sequence

- Sequence of (Modified token, Document ID) pairs.

Assignment Project Exam Help

<https://powcoder.com>

Doc 1

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

Doc 2

Add WeChat powcoder →  
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

# Assignment Project Exam Help

# Indexer steps: Sort

- ## ■ Sort by terms

- And then docID

# Assignment Project Exam Help

<https://powcoder.com>

## Core indexing step

# Add WeChat powcoder

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
the	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	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

# Assignment Project Exam Help

## Indexer steps: Dictionary & Postings

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



Why frequency?  
Will discuss later.

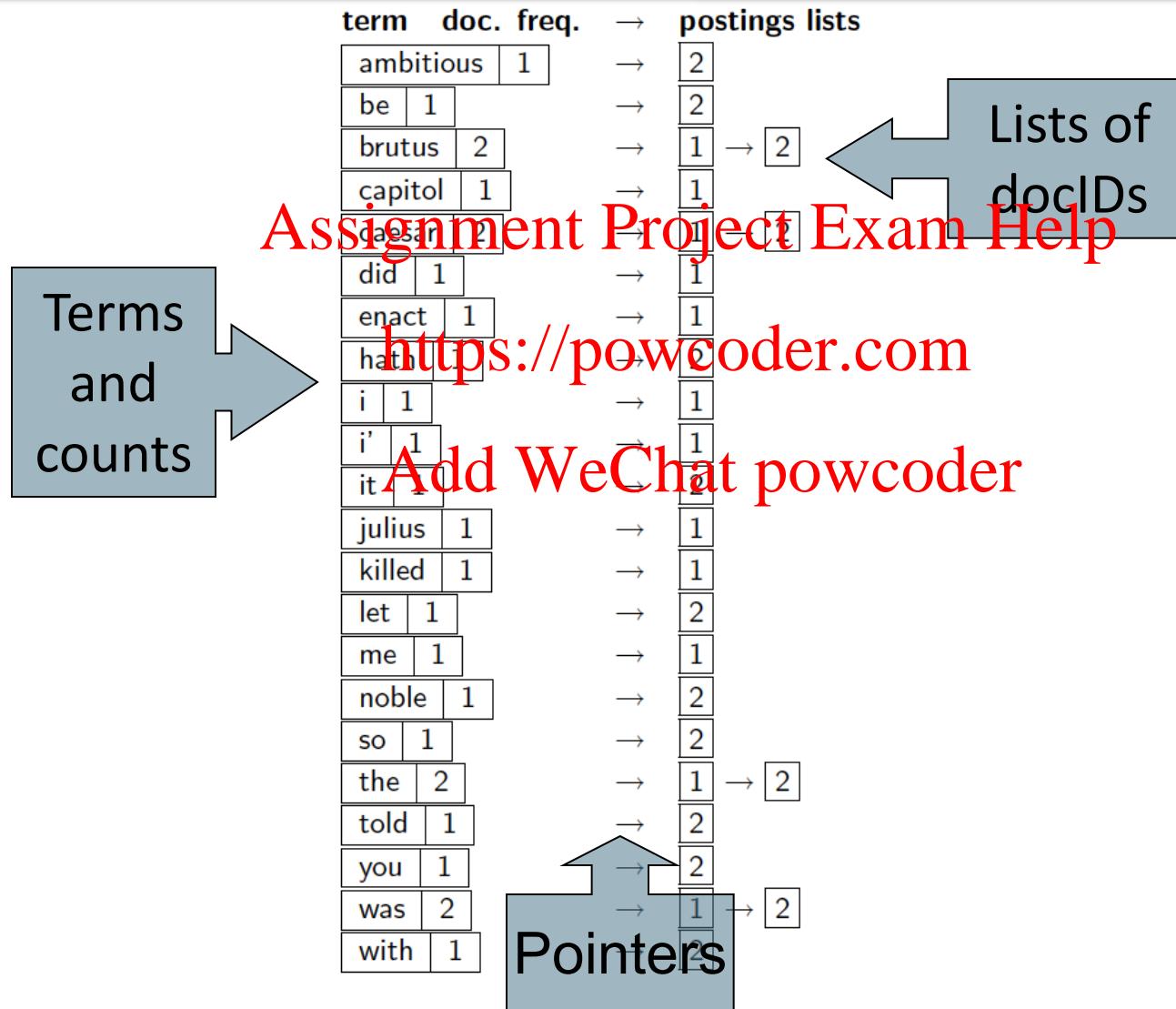
Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



term	doc. freq.	→	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

# Assignment Project Exam Help

## Where do we pay in storage?



# Assignment Project Exam Help

## More on Inverted Index Construction

---

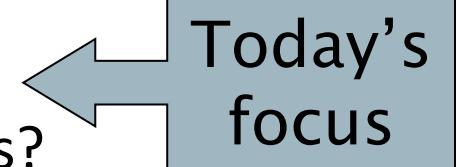
- Hashing-based construction methods are more efficient (though harder to implement)
- Inverted index can be compressed to
  - reduce index size <https://powcoder.com>
  - reduces transfer time between storage and memory

Add WeChat powcoder

# Assignment Project Exam Help

## The index we just built

- How do we process a query?
  - Later - what kinds of queries can we process?



## Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

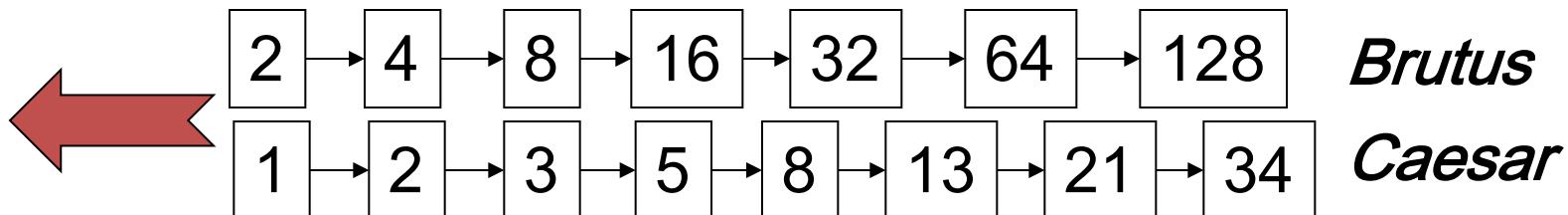
# Assignment Project Exam Help

## Query processing: AND

- Consider processing the query:

***Brutus AND Caesar***

- Locate *Brutus* in the Dictionary;
  - Retrieve its postings,  
<https://powcoder.com>
- Locate *Caesar* in the Dictionary;
  - Retrieve its postings.
- “Merge” the two postings:



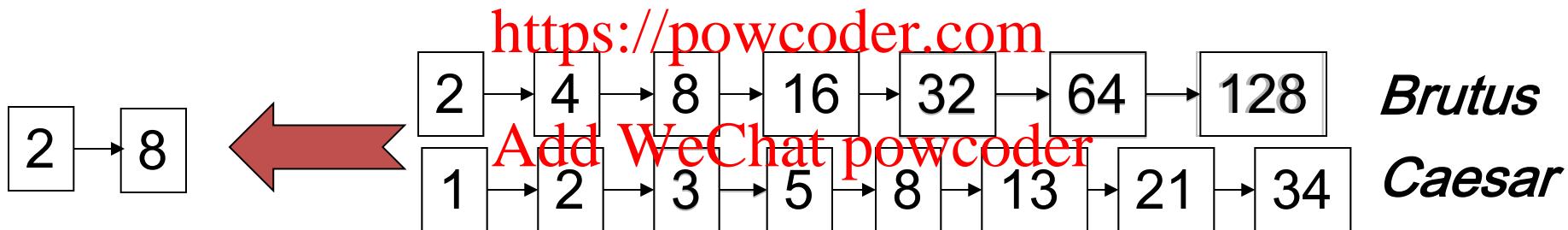
# Assignment Project Exam Help

## The merge

Add WeChat powcoder

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

Add WeChat powcoder



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( $p_1, p_2$ )

```
1  answer ← ⟨ ⟩  
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$   
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$   
4      then ADD(answer,  $\text{docID}(p_1)$ )  
5           $p_1 \leftarrow \text{next}(p_1)$   
6           $p_2 \leftarrow \text{next}(p_2)$   
7      else if  $\text{docID}(p_1) < \text{docID}(p_2)$   
8          then  $p_1 \leftarrow \text{next}(p_1)$   
9          else  $p_2 \leftarrow \text{next}(p_2)$   
10 return answer
```

# Assignment Project Exam Help

## Boolean queries: Exact match

---

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:  
**Assignment Project Exam Help**
  - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms <https://powcoder.com>
    - 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

# Assignment Project Exam Help

Example: Add WeChat <http://www.westlaw.com/>

---

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992) Assignment Project Exam Help
- 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
    - foo! = foo\*, /3 = within 3 words, /S = in same sentence

# Assignment Project Exam Help

Example: Add WeChat powcoder <http://www.westlaw.com/>

---

- Another example query:
  - Requirements for disabled people to be able to access a workplace
  - disabl! /p address! /s work site workplace (employment /3 place)
- 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
- But that doesn't mean it actually works better...

Assignment Project Exam Help  
Boolean queries:

More general merges

---

- Exercise: Adapt the merge for the queries:

***Brutus AND NOT Caesar***

Assignment Project Exam Help

***Brutus OR NOT Caesar***

<https://powcoder.com>

Can we still run through the merge in time  $O(x+y)$ ?

What can we achieve?

# Assignment Project Exam Help

## Merging Add WeChat powcoder

---

What about an arbitrary Boolean formula?

**(Brutus OR Caesar) AND NOT**

**(Antony OR Cleopatra)**

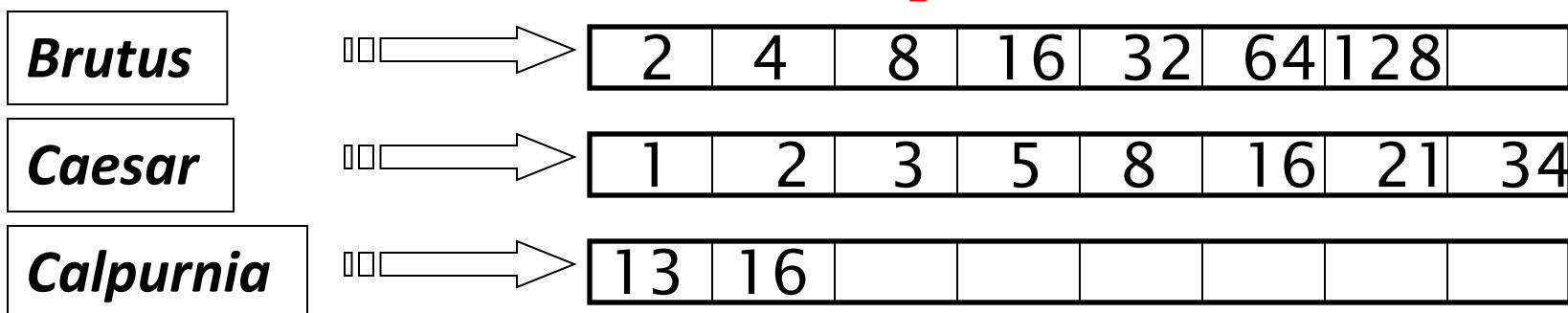
- Can we always merge in “linear” time?
  - Linear in what? Add WeChat powcoder
- Can we do better?

# Assignment Project Exam Help

## 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](https://powcoder.com), then AND them together.

Add WeChat powcoder



Query: **Brutus AND Calpurnia AND Caesar**

# Assignment Project Exam Help

## Query optimization example

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

### Assignment Project Exam Help

This is why we kept  
<https://powcoder.com>  
document freq. in dictionary

### Add WeChat powcoder

<b>Brutus</b>	⇒	2   4   8   16   32   64   128
<b>Caesar</b>	⇒	1   2   3   5   8   16   21   34
<b>Calpurnia</b>	⇒	13   16

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

# Assignment Project Exam Help

## More general optimization

---

- e.g., **(madding OR crowd) AND (ignoble OR strife) AND (light OR lord)**
- 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.

# Assignment Project Exam Help

## Exercise Add WeChat powcoder

- Recommend a query processing order for

Assignment Project Exam Help

*(tangerine OR trees) AND https://powcoder.com*  
*(marmalade OR skies) AND*  
*(kaleidoscope OR eyes)* Add WeChat powcoder

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

**Q:** Any more accurate way to estimate the cardinality of intermediate results?

**Q:** Can we merge multiple lists (>2) simultaneously?

# Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

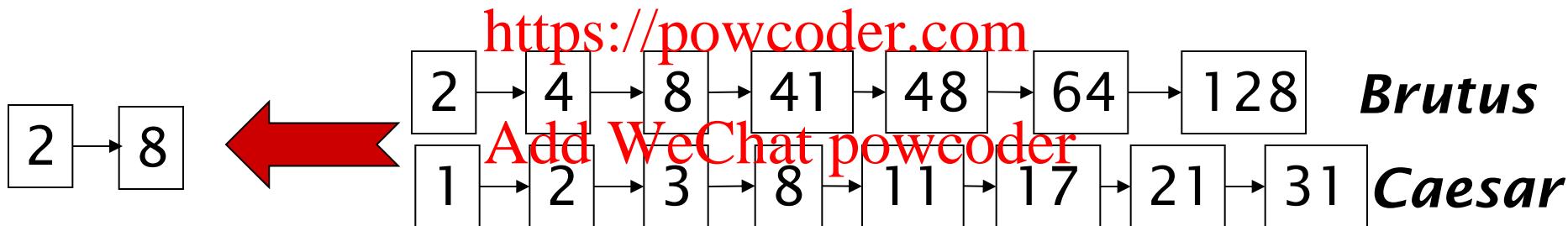
**FASTER POSTINGS MERGES:  
SKIP POINTERS/SKIP LISTS**

# Assignment Project Exam Help

## Recall basic merge

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

Add WeChat powcoder

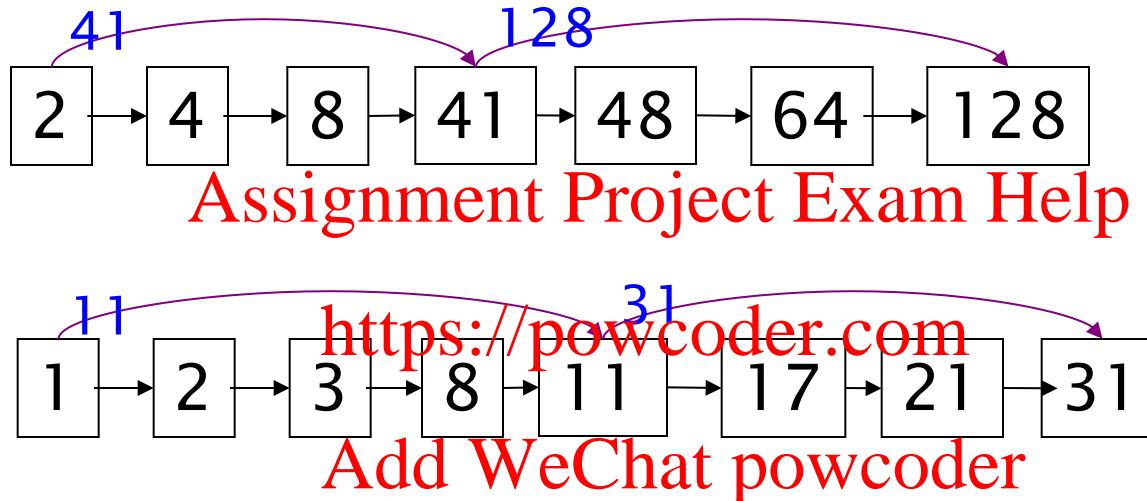


If the list lengths are  $m$  and  $n$ , the merge takes  $O(m+n)$  operations.

Can we do better?

Yes (if index isn't changing too fast).

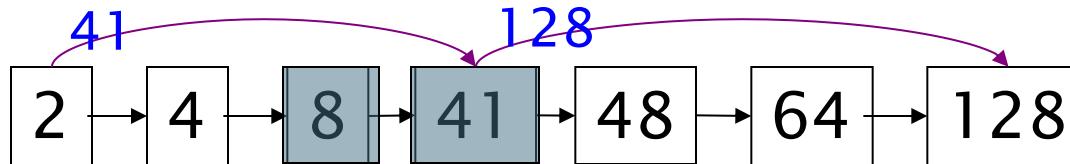
# Augment postings with skip pointers (at indexing time)



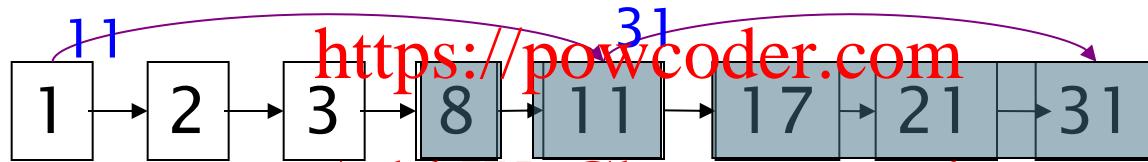
- Why?
- To skip postings that will not figure in the search results.
- How?
- Where do we place skip pointers?

# Assignment Project Exam Help

## Query processing with skip pointers



Assignment Project Exam Help



Add WeChat powcoder

https://powcoder.com

Suppose we've stepped through the lists until we process 8 on each list. We match it and advance.

We then have 41 and 11 on the lower. 11 is smaller.

But the skip successor of 11 on the lower list is 31, so we can skip ahead past the intervening postings.

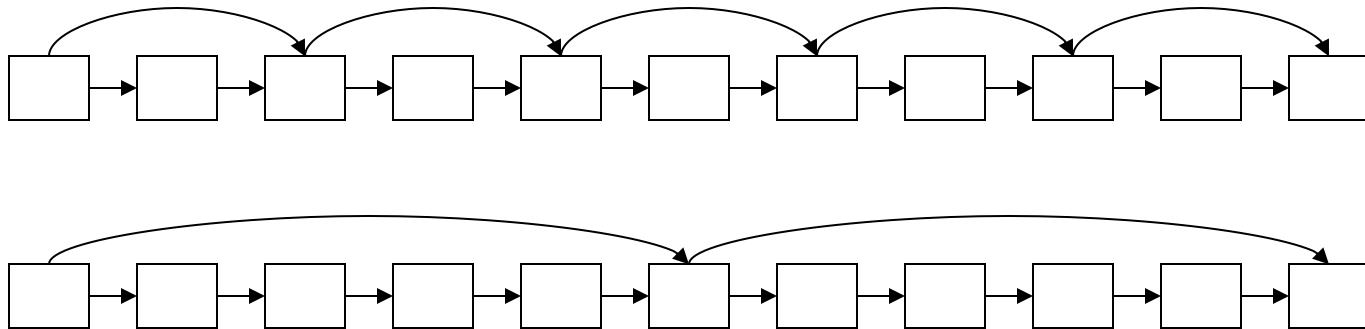
# Assignment Project Exam Help

Can we skip w/o skip pointers?

## Where do we place skips?

- Tradeoff:
  - More skips → shorter skip spans → more likely to skip.  
But lots of comparisons to skip pointers.
  - Fewer skips → few pointer comparison, but then long skip spans → few successful skips.

Add WeChat powcoder



# Assignment Project Exam Help

## Placing skips

Add WeChat powcoder

---

- Simple heuristic: for postings of length  $L$ , use  $L^{1/2}$  evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is <https://powcoder.com>, harder if  $L$  keeps changing because of updates.

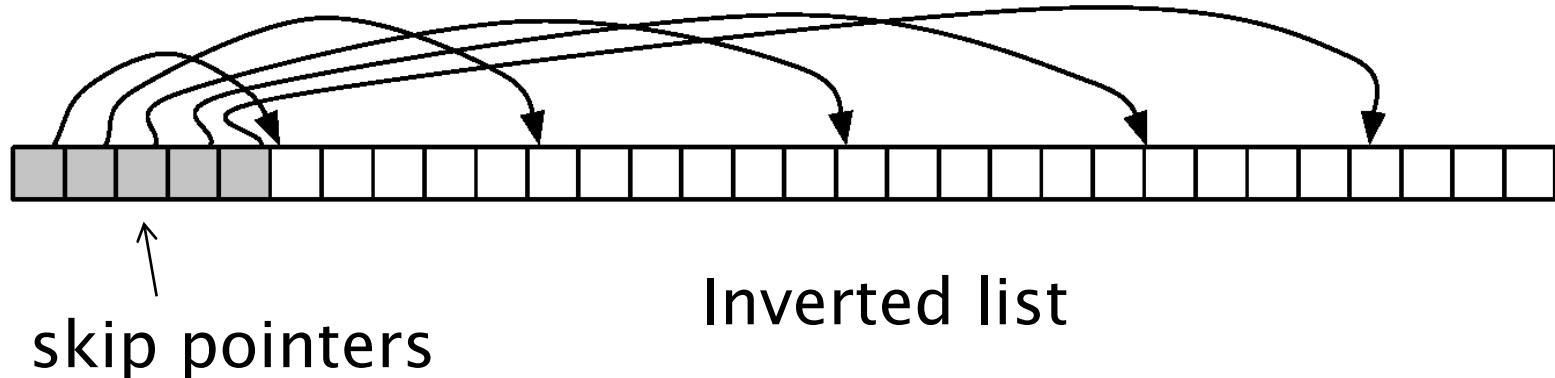
Add WeChat powcoder
- This definitely used to help; with modern hardware it may not (Bahle et al. 2002) unless you're memory-based
  - The I/O cost of loading a bigger postings list can outweigh the gains from quicker in memory merging!

# Skip Pointers

Add WeChat powcoder

- A skip pointer ( $d, p$ ) contains a document number  $d$  and a byte (or bit) position  $p$ 
  - Means there is an inverted list posting that starts at position  $p$ , and the posting before it was for document  $d$

Add WeChat powcoder



# Assignment Project Exam Help

Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## PHRASE QUERIES AND POSITIONAL INDEXES

# Assignment Project Exam Help

## Phrase queries

- Want to be able to answer queries such as “*stanford university*” – as a phrase
- Thus the sentence “*I went to university at Stanford*” is not a match
  - The concept of phrase queries has proven easily understood by users, one of the few “advanced search” ideas that works
  - Many more queries are *implicit phrase queries*
- For this, it no longer suffices to store only  $\langle \text{term} : \text{docs} \rangle$  entries

# Assignment Project Exam Help

## Solution 1: Biword indexes

---

- Index every consecutive pair of terms in the text as a phrase
- For example the text “Friends, Romans, Countrymen” would generate the biwords
  - *friends romans*
  - *romans countrymen*
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate.

# Assignment Project Exam Help

## Longer phrase queries

- Longer phrases are processed as we did with wild-cards:
- *stanford university palo alto* can be broken into the Boolean query on biwords:  
<https://powcoder.com>

*stanford university AND university palo AND palo alto*  
Add WeChat powcoder

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.

Can have false positives!

# Assignment Project Exam Help

## Extended biwords

- Parse the indexed text and perform part-of-speech-tagging (POST).
- Bucket the terms into (say) Nouns (N) and articles/prepositions (X).
- Call any string of terms of the form  $NX^*N$  an extended biword.
  - Each such extended biword is now made a term in the dictionary.
- Example: *catcher in the rye*

N	X	X	N
---	---	---	---
- Query processing: parse it into N's and X's
  - Segment query into enhanced biwords
  - Look up in index: *catcher rye*

# Assignment Project Exam Help

## Issues for biword indexes

---

- False positives, as noted before
- Index blowup due to bigger dictionary
  - Infeasible for more than biwords, big even for them  
<https://powcoder.com>
- Biword indexes are not the standard solution (for all biwords) but can be part of a compound strategy

## Assignment Project Exam Help

### Solution 2<sup>Add WeChat powcoder</sup>: Positional indexes

---

- In the postings, store, for each ***term*** the position(s) in which tokens of it appear:

Assignment Project Exam Help

<***term***, number of docs containing ***term***;

doc1: position1, position2 ... ;

doc2: position1, position2 ... ;

etc.>

# Assignment Project Exam Help

## Positional index example

<*be*: 993427;

1: 7, 18, 33, 72, 86, 231;

2: 3, 149;

4: 17, 191, 291, 430, 434;

5: 363, 367, ...

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Which of docs 1,2,4,5 could contain “*to be or not to be*”?

- For phrase queries, we use a merge algorithm recursively at the document level
- But we now need to deal with more than just equality

# Assignment Project Exam Help

## Processing a phrase query

- Extract inverted index entries for each distinct term:  
*to, be, or, not.*
- Merge the *Assignment Projects Exam Help* all positions with “**to be or not to be**”.  
<https://powcoder.com>
  - *to:*
    - 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...
  - *be:*
    - 1:17,19; 4:17,191,291,430,434; 5:14,19,101; ...
- Same general method for proximity searches

# Assignment Project Exam Help

## Proximity queries

---

- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
  - Again, here,  $/k$  means “within  $k$  words of”.
- Clearly, positional indexes can be used for such queries; biword indexes cannot.  
<https://powcoder.com>
- Exercise: Adapt the linear merge of postings to handle proximity queries. Can you make it work for any value of  $k$ ?
  - This is a little tricky to do correctly and efficiently
  - See Figure 2.12 of IIR (Page 39)
  - There’s likely to be a problem on it!

# Assignment Project Exam Help

## Add WeChat powcoder

```

POSITIONALINTERSECT( $p_1, p_2, k$ )
1   answer  $\leftarrow \langle \rangle$ 
2   while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3   do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4     then  $l \leftarrow \langle \rangle$ 
5      $pp_1 \leftarrow \text{positions}(p_1)$ 
6      $pp_2 \leftarrow \text{positions}(p_2)$ 
7     while  $pp_1 \neq \text{NIL}$ 
8       do while  $pp_2 \neq \text{NIL}$ 
9         do if  $|\text{pos}(pp_1) - \text{pos}(pp_2)| \leq k$ 
10        then ADD( $l, \text{pos}(pp_2)$ )
11        else if  $\text{pos}(pp_2) > \text{pos}(pp_1)$ 
12          then break
13           $pp_2 \leftarrow \text{next}(pp_2)$ 
14        while  $l \neq \langle \rangle$  and  $|l[0] - \text{pos}(pp_1)| > k$ 
15          do DELETE( $l[0]$ )
16          for each  $ps \in l$ 
17            do ADD(answer,  $\langle \text{docID}(p_1), \text{pos}(pp_1), ps \rangle$ )
18           $pp_1 \leftarrow \text{next}(pp_1)$ 
19           $p_1 \leftarrow \text{next}(p_1)$ 
20           $p_2 \leftarrow \text{next}(p_2)$ 
21        else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
22          then  $p_1 \leftarrow \text{next}(p_1)$ 
23          else  $p_2 \leftarrow \text{next}(p_2)$ 
24 return answer

```

**Assignment Project Exam Help**

<https://powcoder.com>

**Add WeChat powcoder**

# Assignment Project Exam Help

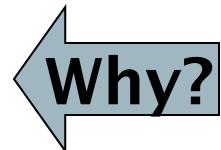
## Positional index size

- You can compress position values/offsets: we'll talk about that in lecture 5
- Nevertheless, a positional index expands postings storage substantially <https://powcoder.com>
- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase and proximity queries ... whether used explicitly or implicitly in a ranking retrieval system.

# Assignment Project Exam Help

## Positional index size

- Need an entry for **each occurrence**, not just once per document
- Index size depends on average document size
  - Average web page has ~1000 terms
  - SEC filings, books, even some epic poems ... easily 100,000 terms
- Consider a term with frequency 0.1%



Document size	Postings	Positional postings
1000	1	1
100,000	1	100

# Assignment Project Exam Help

## Rules of thumb

- A positional index is 2–4 times as large as a non-positional index
- Positional index size 35–50% of volume of original text  
<https://powcoder.com>
- Caveat: all of this holds for “English-like” languages

# Assignment Project Exam Help

## Combination schemes

---

- These two approaches can be profitably combined
  - For particular phrases ("Michael Jackson", "Britney Spears") it is inefficient to keep on merging positional postings lists
    - Even more so for phrases like "The Who"
- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme
  - A typical web query mixture was executed in  $\frac{1}{4}$  of the time of using just a positional index
  - It required 26% more space than having a positional index alone

[Optional]

**Assignment Project Exam Help**

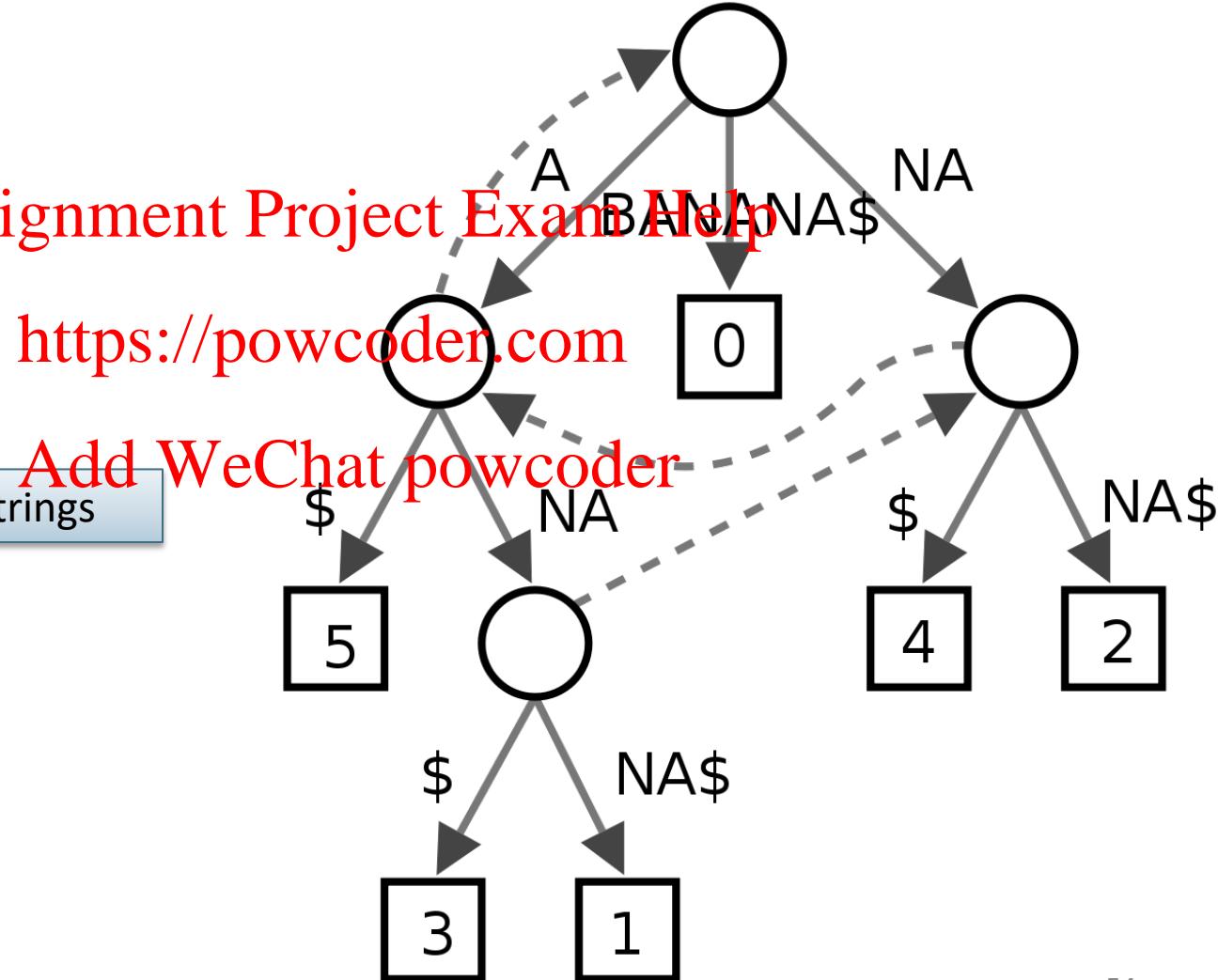
\$ &lt; any char

**Solution 3: Suffix Tree/Array**

- BANANA\$
- BANANA\$ pos:0
- ANANA\$ pos:1
- NANA\$ pos:2
- ANA\$ pos:3
- NA\$ pos:4
- A\$ pos:5

 Sort on the strings

- A\$ pos:5
- ANA\$ pos:3
- ANANA\$ pos:1
- BANANA\$ pos:0
- NA\$ pos:4
- NANA\$ pos:2



[Optional]

# Assignment Project Exam Help

\$ < any char

## Suffix Array

- BANANA\$

- BANANA\$ pos:0
- ANANA\$ pos:1
- NANA\$ pos:2
- ANA\$ pos:3
- NA\$ pos:4
- A\$ pos:5
- \$ pos:6



- \$ pos:6
- A\$ pos:5
- ANAS pos:3
- ANANA\$ pos:1
- BANANA\$ pos:0
- NA\$ pos:4
- NANA\$ pos:2

- If the original string is available, each suffix can be completely specified by the index of its first character

<https://powcoder.com> Index size (incl. data): 5n

Add WeChat powcoder  
Sort on the strings



B	A	N	A	N	A	\$
6	5	3	1	0	4	2

← Suffix array

Binary search (using offsets to fetch the 'key')

# Assignment Project Exam Help

## Resources for today's lecture

---

- *Introduction to Information Retrieval*, chapter 1
- Shakespeare:
  - <http://www.rhymezone.com/shakespeare/>
  - Try the neat browser by keyword sequence feature! <https://powcoder.com>
- *Managing Gigabytes*, chapter 3.2
- *Modern Information Retrieval*, chapter 8.2

# Assignment Project Exam Help

## Resources for today's lecture

---

- Skip Lists theory: Pugh (1990)
  - Multilevel skip lists give same  $O(\log n)$  efficiency as trees
- H.E. Williams, J. Zobel, and D. Bahle, 2004, "Fast Phrase Querying with Combined Indexes", ACM Transactions on Information Systems  
<https://powcoder.com>  
<http://www.seg.rmit.edu.au/research/research.php?author=4>
- D. Bahle, H. Williams and J. Zobel, Efficient phrase querying with an auxiliary index. SIGIR 2002, pp. 215-221.