CS 50: Software Design and Implementation

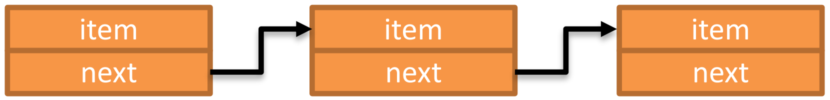
Tiny Search Engine Indexer

Agenda

1. TSE component review
2. Indexer
3. Activity

Recall Bags, Sets, Hashtables, and Counters from Lab 3

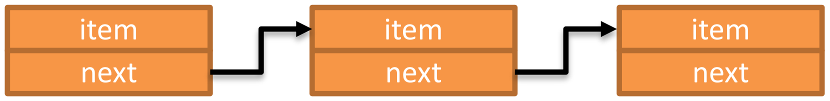
Bag (duplicates allowed)



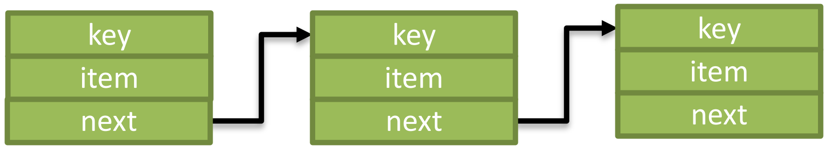
	stores item	***uses key***	***items ordered***	***retrieval***	***insertion of duplicates***
bag	yes	no	no	any item	allowed

Recall Bags, Sets, Hashtables, and Counters from Lab 3

Bag (duplicates allowed)



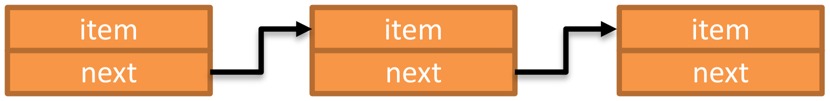
Set (duplicates not allowed)



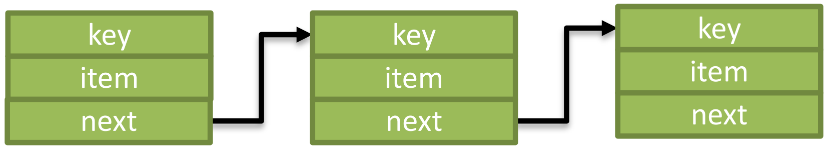
	stores item	***uses key***	***items ordered***	***retrieval***	***insertion of duplicates***
bag	yes	no	no	any item	allowed
set	yes	yes	no	by key	error

Recall Bags, Sets, Hashtables, and Counters from Lab 3

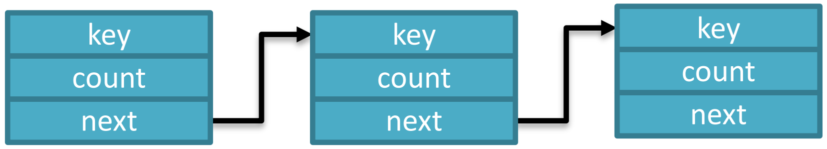
Bag (duplicates allowed)



Set (duplicates not allowed)



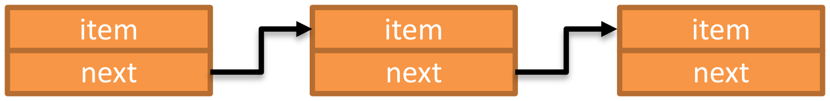
Counters (duplicates increment count)



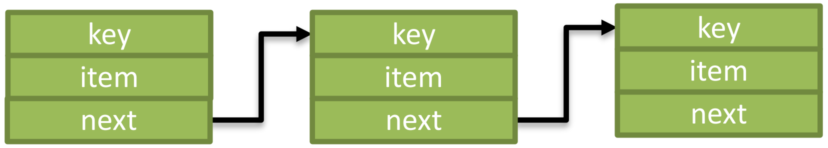
	stores item	*uses key*	*items ordered*	*retrieval*	*insertion of duplicates*
bag	yes	no	no	any item	allowed
set	yes	yes	no	by key	error
counters	no	yes	no	by key	increments count

Recall Bags, Sets, Hashtables, and Counters from Lab 3

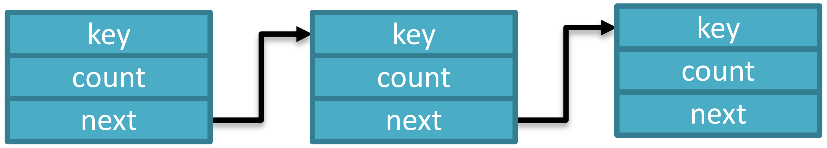
Bag (duplicates allowed)



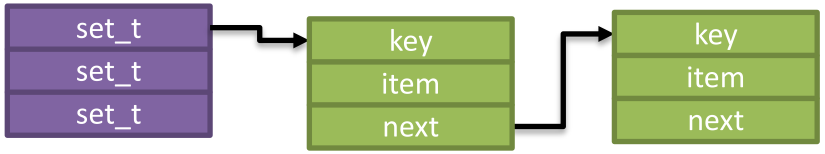
Set (duplicates not allowed)



Counters (duplicates increment count)



Hash table (duplicates not allowed)



	stores item	***uses key***	***items ordered***	***retrieval***	***insertion of duplicates***
bag	yes	no	no	any item	allowed
set	yes	yes	no	by key	error
counters	no	yes	no	by key	increments count
hashtable	yes	yes	no	by key	error

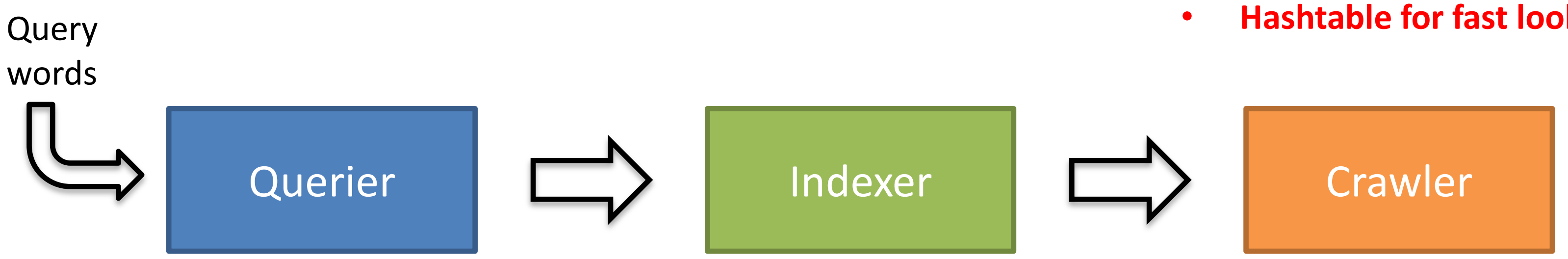
Crawler finds pages reachable from seedURL and stores URL, depth, HTML

Goal:

- **Keep track of to crawl pages**
- **Do not revisit pages**

ADTs?

- **Bag to track pages to see**
- **Hashtable for fast look up of pages seen**



If your crawler didn't work well, find example output at: $loc/tse/tse-output

Use these examples as a source for your indexer

Reference these files, no need to make your own copy

Given:

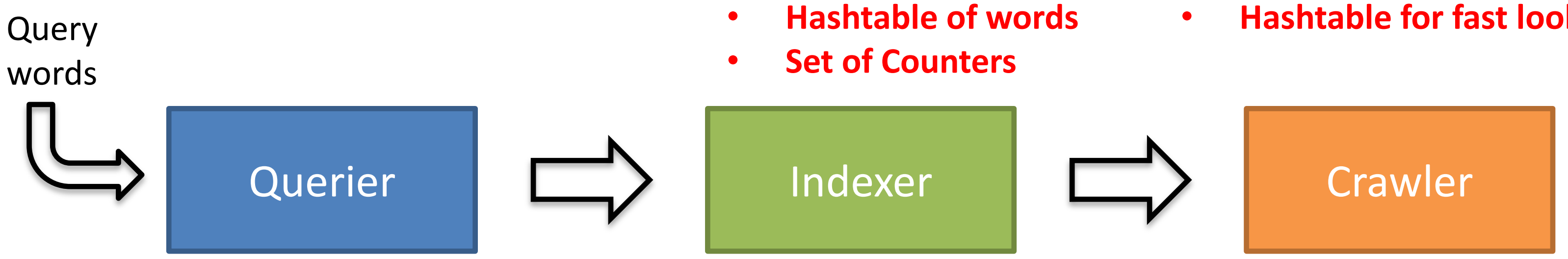
- seedURL
- directory to store results
- depth to search

Follow links to find all reachable pages from seedURL < depth

Store in a separate file for each page in given directory

- URL
- Depth
- HTML

Indexer uses crawler's results and builds data structure to find pages with words



Goal:
- **Fast look up of documents containing a given word**

ADTs?
- **Hashtable of words**
- **Set of Counters**

Given:
- crawler's files

Parse each page's HTML from crawled web pages and discover all words not inside HTML tags

Build data structure to find all documents that contain each discovered word and how many times those words appear

Goal:
- **Keep track of to crawl pages**
- **Do not revisit pages**

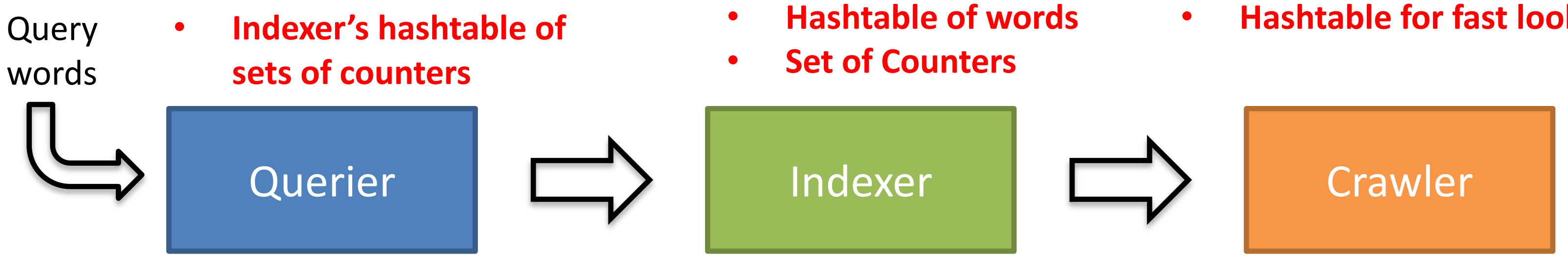
ADTs?
- **Bag to track pages to see**
- **Hashtable for fast look up of pages seen**

Given:
- seedURL
- directory to store results
- depth to search

Follow links to find all 'internal' reachable pages from seedURL < depth

Store in a separate file for each page in given directory
- URL
- Depth
- HTML

Querier finds and ranks pages containing query words



Goal:

- **Fast ranked list of documents containing query words**

ADTs?

- **Indexer's hashtable of sets of counters**

Given:

- Indexer's data structure
- Query words

Find web pages containing query words

Rank pages based on how many times each word appears

Consider AND and OR logic

Goal:

- **Fast look up of documents containing a given word**

ADTs?

- **Hashtable of words**
- **Set of Counters**

Given:

- crawler's files

Parse each page's HTML from crawled web pages and discover all words not inside tags

Build data structure to find all documents that contain each discovered word and how many times those words appear

Save data structure

Goal:

- **Keep track of to crawl pages**
- **Do not revisit pages**

ADTs?

- **Bag to track pages to see**
- **Hashtable for fast look up of pages seen**

Given:

- seedURL
- directory to store results
- depth to search

Follow links to find all 'internal' reachable pages from seedURL < depth

Store in a separate file for each page in given directory

- URL
- Depth
- HTML

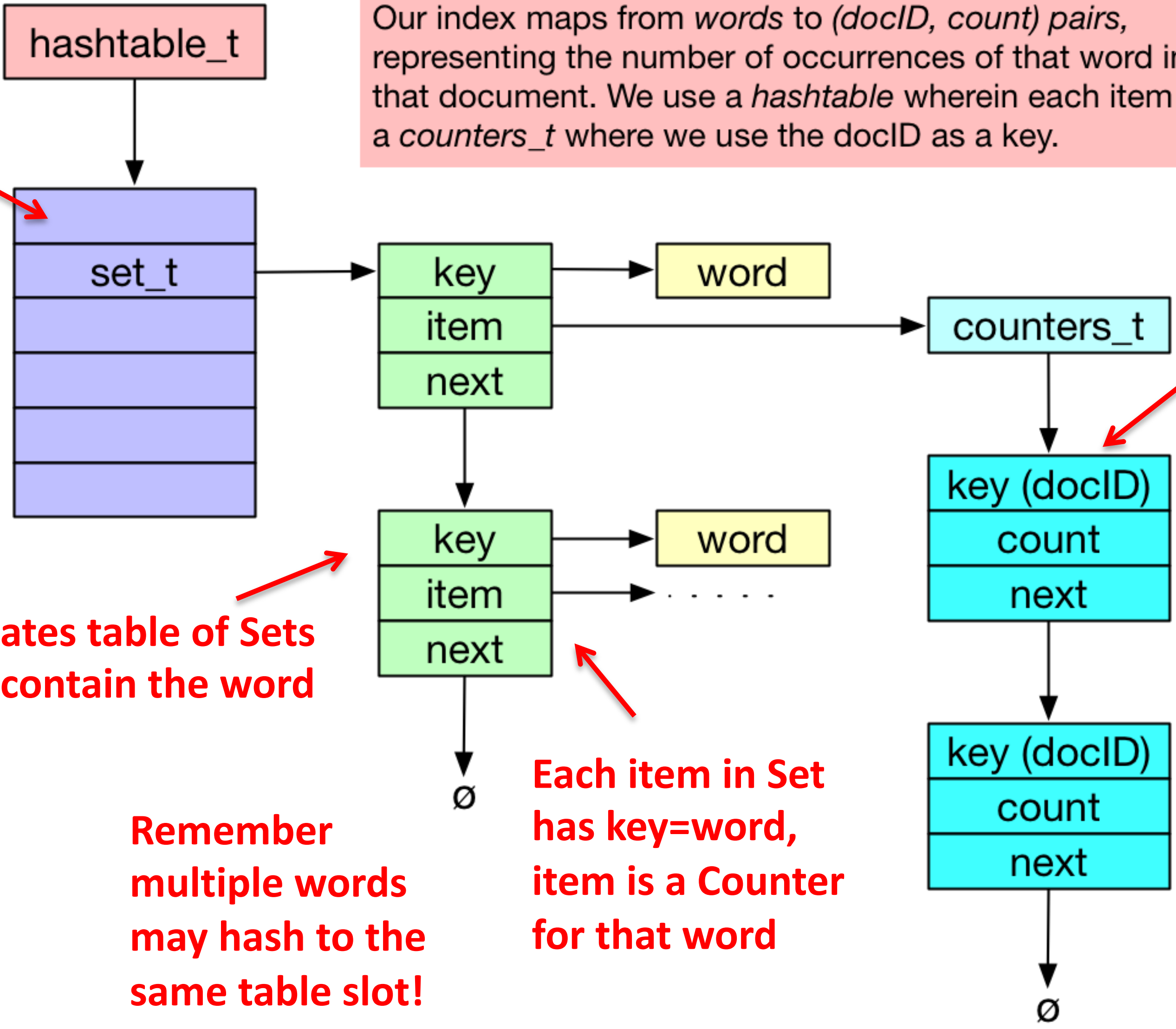
Agenda

1. TSE component review
2. Indexer
3. Activity

The indexer builds a Hashtable of Sets of Counters!

Index data structure

Our index maps from *words* to *(docID, count) pairs,* representing the number of occurrences of that word in that document. We use a *hashtable* wherein each item is a *counters_t* where we use the docID as a key.



Keys are words from HTML pages where words are not inside HTML tags

Hashtable creates table of Sets of pages that contain the word

Remember multiple words may hash to the same table slot!

Each item in Set has key=word, item is a Counter for that word

Each Counter has key=docID and a count of how many times the word appears in the document

One Counter for each document that contains word

Indexer demo: check provided crawler output

Crawler output from our test sites provided in directories here (in your crawler didn't work well)

```
$ ls $loc/tse/tse-output/
cs50-index-0     letters-depth-3/ letters-index-1   letters-index-5   toscrape-index-0   wikipedia-depth-0/  wikipedia-index-1
index.html       letters-depth-4/ letters-index-2   letters-index-6   toscrape-index-1   wikipedia-depth-1/  wikipedia-index-1.s
letters-depth-0/ letters-depth-5/ letters-index-3   toscrape-depth-0/ toscrape-index-1~  wikipedia-depth-2/  wikipedia-index-2
letters-depth-1/ letters-depth-6/ letters-index-3.s toscrape-depth-1/ toscrape-index-2   wikipedia-index-0   wikipedia-index-2.s
letters-depth-2/ letters-index-0  letters-index-4   toscrape-depth-2/ toscrape-index-2.s wikipedia-index-0.s
```

Directory tells website and depth

```
$ ls $loc/tse/tse-output/letters-depth-6
1 2 3 4 5 6 7 8 9
```

Crawler found 9 sites in letters using depth 6

```
$ vi $loc/tse/tse-output/letters-depth-6/1
```

```
http://cs50tse.cs.dartmouth.edu/tse/letters/index.html
0
<html>
<title>home</title>
This is the home page for a CS50 TSE playground.
<a href=A.html>A</a>
</html>

```

Crawler output for page 1

Indexer demo: run indexer

```
$ ./indexer  $loc/tse/tse-output/letters-depth-6 letters.index
  1 Loaded: http://cs50tse.cs.dartmouth.edu/tse/letters/index.html
  1 Indexing page: http://cs50tse.cs.dartmouth.edu/tse/letters/index.html
  1 Add word to index: home
  1 Inc word count: home
  1 Add word to index: this
  1 Inc word count: this
  1 Add word to index: the
  1 Inc word count: the
  1 Inc word count: home
  1 Add word to index: page
  1 Inc word count: page
  1 Add word to index: for
  1 Inc word count: for
  1 Add word to index: tse
  1 Inc word count: tse
  1 Add word to index: playground
  1 Inc word count: playground
  2 Loaded: http://cs50tse.cs.dartmouth.edu/tse/letters/A.html
<snip>
```

File to save output

Crawler directory

Indexer demo: run indexer

```
$ ./indexer  $loc/tse/tse-output/letters-depth-6 letters.index
  1 Loaded: http://cs50tse.cs.dartmouth.edu/tse/letters/index.html
  1 Indexing page: http://cs50tse.cs.dartmouth.edu/tse/letters/index.html
  1 Add word to index: home
  1 Inc word count: home
  1 Add word to index: this
  1 Inc word count: this
  1 Add word to index: the
  1 Inc word count: the
  1 Inc word count: home
  1 Add word to index: page
  1 Inc word count: page
  1 Add word to index: for
  1 Inc word count: for
  1 Add word to index: tse
  1 Inc word count: tse
  1 Add word to index: playground
  1 Inc word count: playground
  2 Loaded: http://cs50tse.cs.dartmouth.edu/tse/letters/A.html
<snip>
```

```
$ cat $loc/tse/tse-output/letters-depth-6/1
http://cs50tse.cs.dartmouth.edu/tse/letters/index.html
0
<html>
<title>home</title>
This is the home page for a CS50 TSE playground.
<a href=A.html>A</a>
</html>
```

File 1

URL

Process each word in HTML (no need to fetch page), add to hashtable with counter for word

Move on to file 2

Indexer demo: view index created

```
$ ./indexer $loc/tse/tse-output/letters-depth-6 letters.index
… output…
$ vi letters.index
```

File to save output

Crawler directory

```
playground 1 1
page 1 1
coding 6 1
this 1 1
home 1 2 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 1
depth 8 1
eniac 4 1
the 1 1
for 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 1
breadth 3 1
tse 1 1
biology 9 1
first 3 1 8 1
search 3 1 8 1
huffman 6 1
traversal 5 1
transform 7 1
fourier 7 1
graph 5 1
algorithm 2 1
fast 7 1
computational 9 1
```

Format:
word docID count [docID count] …
Example: home appears on document 1 two times

indextest.c asks you to:

- **Read an index from a file**
- **Write it back out to a file**
- **Check to see if the results are the same (you'll need to read an index for the querier)**

Indexer high-level pseudo code

High-level pseudo code

1. Validate parameters (pageDirectory and output filename)
2. Read documents from the pageDirectory created by crawler where:
    - the document id starts at 1 and increments by 1 for each new page found by crawler
    - Filename is of form pageDirectory/documentID,
    - First line of the file is the URL, second line of the file is the depth, rest of the file is the page content (the HTML, unchanged)
3. Parse words not inside tags in each page's HTML
4. Build an inverted-index data structure mapping from *words* to *(documentID, count)* pairs, where each *count* represents the number of occurrences of the given word in the given document
5. Write the index to a file (the querier will load this file in Lab 6)

The indexer *may assume* that

- pageDirectory has files named 1, 2, 3, …, without gaps.
- The content of files in pageDirectory follow the format as defined in the crawler specs; thus your code (to read the files) need not have extensive error checking

Agenda

1. TSE component review
2. Indexer
3. Activity