Milestone 3: using the index to search

Once you're indexing documents correctly, you're ready to implement the actual search functionality in search(index, query)! Many people find this part of the problem much easier than building an index. You should use your code from the first milestone, common(), to help out here.

This function is passed the following parameters:

* index is the index produced by your create\_index() function
* query is a string representing the user's query. All the letters in this string are guaranteed to be lowercase, and you can assume the user will never enter punctuation characters as part of the query string (though the starter code doesn't actually strip out punctuation, in case you wanted to do some sort of extension where punctuation characters might be meaningful in how the search is conducted)

search(index, query) should *return a list of the names of the files that contain all the terms in the given query.* Recall that in index, the value associated with each term (the key in the dictionary) is a list of files that contain the given term. This list of files is called the "posting list" for the term. In order to determine which files contain all the terms in a query, start with the posting list for the first term in the query. Then loop over all of the rest of the terms in the query, and consecutively consider the overlap (the *common* elements) of the posting list associated with each subsequent term. When you've processed all the terms in the query this way, the posting list you have left should contain only the files that contain every term in the query. Restated, here's the pseudocode:

posting list = access posting list for first term in query

for each term in the rest of the query (after the first word):

new posting list = access posting list for new term

posting list = common elements between current posting list and new posting list

return posting list

If you were to build an index on the small dataset, your search function should return these results:

Example 1. Calling: search(index, 'apple')

Should produce the list: ['small/1.txt', 'small/2.txt', 'small/3.txt']

Example 2. Calling: search(index, 'ball')

Should produce the list: ['small/1.txt', 'small/3.txt']

Example 3. Calling: search(index, 'lizard')

Should produce the list: ['small/3.txt']

Example 4. Calling: search(index, 'apple ball')

Should produce the list: ['small/1.txt', 'small/3.txt']

Example 5. Calling: search(index, 'dog ball')

Should produce the list: ['small/1.txt']

Example 6. Calling: search(index, 'dog ball hamster')

Should produce the list: []

Example 7. Calling: search(index, 'nope')

Should produce the list: []

Doctests for search() are provided; run them using the Check button.

You can also run your search function and interactively query your search engine by running searchengine.py, specifying the directory name for a set of text files that you'd like to index, and then adding -s at the end of the command line to indicate "search mode". For example, to run your search engine on the small dataset, you'd type python searchengine.py small -s into your terminal; to run it on the BBC dataset you'd type python searchengine.py bbc -s.

Here's the output of a working program running on some sample queries on the BBC News data (user input in ***bold italics***); check if your program produces the same output (note the order of the articles printed might be different, but the set of articles that match each query should be the same as in this sample run):

Query (empty query to stop): ***stanford***

Results for query 'stanford':

1. Title: Yahoo celebrates a decade online, File: bbcnews/066.txt

2. Title: Google to scan famous libraries, File: bbcnews/217.txt

Query (empty query to stop): ***bike***

Results for query 'bike':

1. Title: Games help you 'learn and play', File: bbcnews/291.txt

2. Title: The Force is strong in Battlefront, File: bbcnews/339.txt

Query (empty query to stop): ***stanford bike***

Results for query 'stanford bike':

No results match that query.

Query (empty query to stop): ***windows virus security patch***

Results for query 'windows virus security patch':

1. Title: Microsoft releases patches, File: bbcnews/003.txt

2. Title: Microsoft releases bumper patches, File: bbcnews/162.txt

Query (empty query to stop): ***cheap apple products***

Results for query 'cheap apple products':

1. Title: Apple Mac mini gets warm welcome, File: bbcnews/033.txt

Query (empty query to stop): ***\*user presses "enter" to end program\****

Congratulations, you've just built a working search engine! That's extremely impressive, given you might not have known a lot about programming just 5 weeks ago. It's also a testament to how powerful a tool programming can be -- a little knowledge can go a long way in building some really useful software. Happy searching!