Project: Search Engine

Implement the simplified Search Engine described in Section 23.5.4 for the pages of a small Web site. Use all the words in the pages of the site as index terms, excluding stop words such as articles, prepositions, and pronouns.

Submit the following four files:

1. A read me file that contains details of your approach to the problem, including description of Algorithms and Data Structure used.

2. Your coded, well-commented code file in your favorite language, such as Python, Java, C++,...

3. The input file that contains the few pages you have used as input, including some links to your other pages..

4. Output file that has samples of your run. Make sure you have tested the boundary conditions.

5. Provide a short video demonstrating the execution of your project. (Include the testing of boundary conditions as well).

The following Question and Answers should be used as further guidance on your implementation:

1. Should I use internet documents or create my own?

Answer: You may download several pages (5 to 10) from Internet, or develop them yourself, and use them as input. Make sure there are some hyperlink between the documents and you can add more pages to your existing ones that is gathered by Web Crawler.

2. Can I use Other Search Engine or Data Structures?

Answer: No. You shall implement what is described in Section 23.5.4. using the approach specified. You are not allowed to use Algorithms and Data Structures that are not covered in the textbook, but you can develop new algorithms based on the data structures that you have seen and proved in the textbook.

3. What criteria should I use for ranking?

Answer: I leave that to you to come up with some simple criteria and algorithm to implement ranking, such as the number of times a word has appeared in the document or any other idea you have. But please explain your approach.

4. Can I provide screenshots of my output?

Answer: Yes.

5. How can I read the web pages in?

Answer:

If you are using java, use jsoup: [https://jsoup.org/ (Links to an external site.)Links to an external site.](https://jsoup.org/).  
  
If you are using Python, use beautiful soup:https://www.crummy.com/software/BeautifulSoup/bs4/doc/

6. Can I use any available packages for parsing?

Answer: Yes

23.5.4 Search Engines

The World Wide Web contains a huge collection of text documents (web pages).

Informationaboutthesepagesisgatheredbyaprogramcalledawebcrawler, which

then stores this information in a special dictionary database. A web search engine

allows users to retrieve relevant information from this database, thereby identifying

relevant pages on the web containing given keywords. In this section, we present a

simplified model of a search engine.

Inverted Files

The core information stored by a search engine is a dictionary, called an inverted

index or inverted file, storing key-value pairs (w,L), where w is a word and L

is a collection of references to pages containing word w. The keys (words) in

this dictionary are called index terms and should be a set of vocabulary entries

and proper nouns as large as possible. The elements in this dictionary are called

occurrence lists and should cover as many web pages as possible.

We can efficiently implement an inverted index with a data structure consisting

of the following:

• An array storing the occurrence lists of the terms (in no particular order)

• A compressed trie for the set of index terms, where each external node stores

the index of the occurrence list of the associated term.

The reason for storing the occurrence lists outside the trie is to keep the size of the

trie data structure sufficiently small to fit in internal memory. Instead, because of

their large total size, the occurrence lists have to be stored on disk.

With our data structure, a query for a single keyword is similar to a word match-

ing query (see Section 23.5.1). Namely, we find the keyword in the trie and we

return the associated occurrence list.

When multiple keywords are given and the desired output is the pages contain-

ing all the given keywords, we retrieve the occurrence list of each keyword using

the trie and return their intersection. To facilitate the intersection computation,

each occurrence list should be implemented with a sequence sorted by address or

with a dictionary, which allows for a simple intersection algorithm similar to sorted

sequence merging (Section 8.1).

In addition to the basic task of returning a list of pages containing given key-

words, search engines provide an important additional service by ranking the pages

returned by relevance. Devising fast and accurate ranking algorithms for search

engines is a major challenge for computer researchers and electronic commerce

companies

8个web读进来成数组 ；数组第一项存网址？，存第几项开始，和count

每项存入 A\_80

所有title 建树 （除掉stop word）