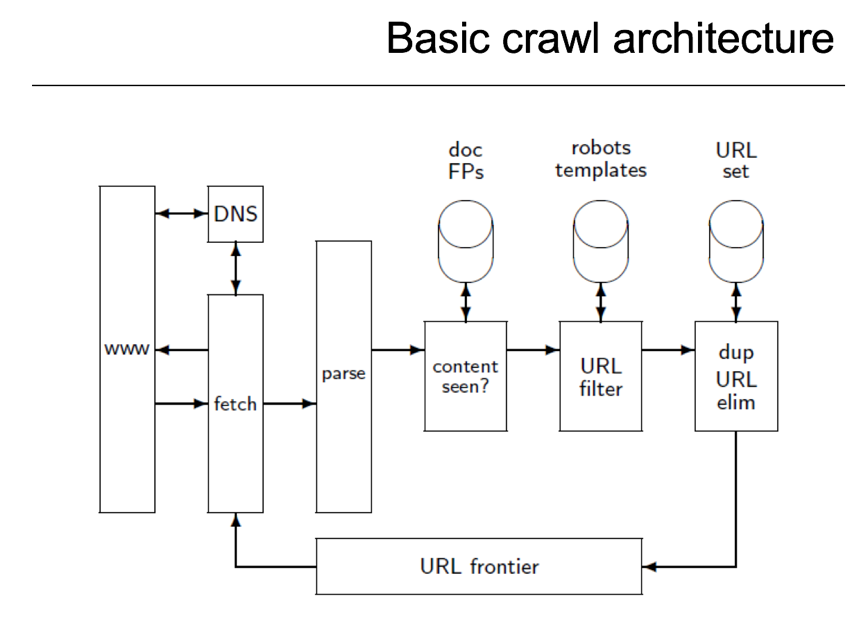
Project Questions

1. Implement your crawler according to requirements. Describe your major data structures. Identify the key properties of a web crawler. Describe in detail how each of these properties is implemented in your code. [25 points]

**Major Data Structures:**

* ***Queue*: Used for the URL frontier (best choice for a FIFO data structure that is constantly having new URLs added and crawled URLs being popped off)**
* ***Nested Dictionary*: Used for the term-frequency matrix (most intuitive implementation of a 2-dimensional matrix in Python)**
* ***Hash Map*: Used for storing hashes of the contents of visited webpages for the purpose of (exact) duplicate detection. The hash of a page’s contents points to the page’s URL ie. {hash:url}**

**Key Properties:**

* ***Robustness*: The crawler utilizes error catching, input validation, and edge-case handling for broken links, duplicates (URLs and page contents), and stale URLs (checks freshness of already crawled URLs to see if they should be crawled again).**
* ***Politeness*: The crawler has a default crawl delay of 2 seconds between each web page that it crawls. However, the crawler looks for a robots.txt file in the web root before crawling the site, which, if a crawl-delay attribute is present, will set its crawl delay accordingly.**
* ***Scalability*: The crawler was built with scalability in mind. Here are a few examples:**
  + **The crawler utilizes functional programming where advantageous, allowing for easy implementation of parallelism if desired.**
  + **The program accommodates the use of a seed set of variable length throughout the whole program.**
  + **The crawler is split into many python files, each of which implement a single class with member functions that corresponds to a part of the crawl architecture. It would require relatively minimal overhead to have different parts of the program over multiple machines.**
* ***Architecture*: The crawler follows an object-oriented approach to the “Basic Crawler Architecture” diagram we looked at in class. Each python file in the project source implements a class that corresponds to an entity in this diagram (with a couple differences). *DETAILED EXPLANATION OF EACH CLASS CAN BE FOUND IN FILE CALLED “CrawlerArchitectureDetailed.docx” (included in both Canvas submission and project directory)***

1. Use your crawler to list the URL of all pages in the test data and report all out-going links of the test data (i.e. items you must not crawl). [10 points]
   * **http://lyle.smu.edu**
   * **http://lyle.smu.edu**
   * **http://oracle.com.edgesuite.net/timeline/oracle/**
   * **http://www.gedpage.com/soundex.html**
   * **http://www.smu.edu/EnrollmentServices/Registrar/Enrollment/FinalExamSchedule/Spring2018**
   * **https://smu.instructure.com**
   * **http://lyle.smu.edu**
   * **http://lyle.smu.edu**
   * **http://tartarus.org/~martin/PorterStemmer/**
   * **http://en.wikipedia.org/wiki/Document\_classification**
   * **http://en.wikipedia.org/wiki/Stop\_words**
   * **http://en.wikipedia.org/wiki/Tf\*idf**
   * **http://lucene.apache.org/core/**
   * **http://search.carrot2.org/stable/search**
   * [**http://9ol.es/porter\_js\_demo.html**](http://9ol.es/porter_js_demo.html)

**Proof:**

1. Implement exact duplicate detection, and report if any URLs refer to already seen content. [10 points]
   * [**https://lyle.smu.edu/~fmoore/index\_duplicate.htm**](https://lyle.smu.edu/~fmoore/index_duplicate.htm)

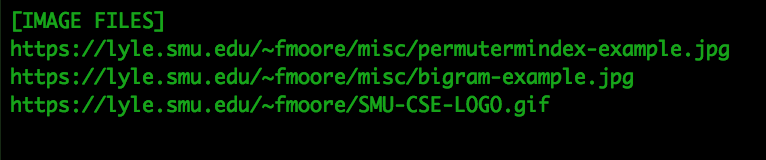
**Proof:**

****

1. Use your crawler to list all broken links within the test data. [10 points]
   * **https://lyle.smu.edu/~fmoore/mailto:fmoore@smu.edu**
   * **https://lyle.smu.edu/~fmoore/mailto:fmoore@lyle.smu.edu**
   * **https://lyle.smu.edu/~fmoore/syl\_5330.pdf**
   * **https://lyle.smu.edu/~fmoore/syl\_7330.pdf**
   * **https://lyle.smu.edu/~fmoore/does\_not\_exist.htm**
   * **https://lyle.smu.edu/~fmoore/misc/count\_letters.txt**
   * **https://lyle.smu.edu/~fmoore/misc/count\_letters\_duplicate.txt**
   * **http://lyle.smu.edu/~fmoore/this\_aint\_gonna\_work.htm**
   * [**https://lyle.smu.edu/~fmoore/this\_aint\_gonna\_work.htm**](https://lyle.smu.edu/~fmoore/this_aint_gonna_work.htm)

**Proof:**

1. List the URLs of graphic (gif, jpg, jpeg, png) files are included in the test data. [10 points]
   * **https://lyle.smu.edu/~fmoore/misc/permutermindex-example.jpg**
   * **https://lyle.smu.edu/~fmoore/misc/bigram-example.jpg**
   * [**https://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif**](https://lyle.smu.edu/~fmoore/SMU-CSE-LOGO.gif)

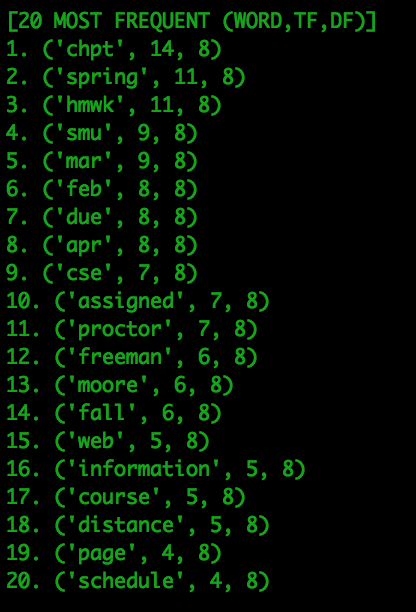
**Proof:**

1. Your crawler must save the words from each page of type (.txt, .htm, .html, .php). Make sure that you do not save HTML markup. A word is a string of non-space characters, beginning with an alphabetic character. It may contain special characters, but the last character of a word is either alphabetic or numeric. Perform case insensitive matching.

In this process, give each page a unique document ID. The output of this step will be a term-document frequency matrix. Your program may generate the data to be further processed in a spreadsheet (Excel or equivalent). [25 points]

**Output matrix is outputted as a csv file called “term-frequency.csv”**

1. Report the 20 most common words with its document frequency. [10 points]

**# , Word , tf , df**

**1. 'chpt', 14, 8**

**2. 'spring', 11, 8**

**3. 'hmwk', 11, 8**

**4. 'smu', 9, 8**

**5. 'mar', 9, 8**

**6. 'feb', 8, 8**

**7. 'due', 8, 8**

**8. 'apr', 8, 8**

**9. 'cse', 7, 8**

**10. 'assigned', 7, 8**

**11. 'proctor', 7, 8**

**12. 'freeman', 6, 8**

**13. 'moore', 6, 8**

**14. 'fall', 6, 8**

**15. 'web', 5, 8**

**16. 'information', 5, 8**

**17. 'course', 5, 8**

**18. 'distance', 5, 8**

**19. 'page', 4, 8**

**20. 'schedule', 4, 8**

(END)