Programming Assignment 2: Subject-Specific Web Crawler

Assigned: Feb. 22
Due: Mar. 21

The object of this project is to write a crawler that downloads pages dealing with a particular subject.

Specification

Input

- A URL from which to start the crawl (presumably, a web page that deals with the subject).
- A query: a set of words, possibly null.
- A path name for a directory to save the downloaded pages.
- A maximum number of pages to download. This should default to 50.
- A flag for generating a trace. This defaults to `false`.

These should be input as flagged command line arguments. The URL is flagged by "-u", the query by "-q", the path by "-docs" the maximum number of pages by "-m", the trace flag by "-t".

So a typical query might be
```
java crawler -u https://en.wikipedia.org/wiki/Cleopatra -q Cleopatra Caesar Anthony -docs ~/myaccount/cleopatra/ -m 300
```

Functionality:

- Each of the downloaded pages should be saved to a directory.
- Output a trace to standard output if specified.

Search

Overall, you want to do a best-first search through the space of web pages. To do that, you should maintain a priority queue of URLs to be downloaded and a set of URLs that you have already downloaded. (Once a URL has been downloaded, it doesn't go back onto the priority queue).

The overall algorithm is this:

```
crawler(StartingURL, Query, MaxPages, Debug) {
   Q = new FIFOQueue(StartingURL);
   Seen = emptyset;
   repeat {
       L = pop the highest scored URL from Q;
       if (Debug) println("\n Downloading:", L, ". Score = " L.score);
       issue an request for L;
       if (the request successfully returns HTML page P) {
           add L to Seen;
           add P to the directory;
           if (Size(Seen) >= MaxPages) exit;
           if (Debug) println("Received:", L);
           Links = set of all links in P;
           for (each link M in Links) {
              S = Score(M,P,Query);
              U = URL of M;
              if (U is not in Seen) {
                 if (U is not in Q) {
                      add U to Q with score S;
                      if (Debug) println("Adding to queue: ", U
                                         " with score " S);
                    }
                 else {
                    add S to the score of U in Q;
                    if (Debug) println("Adding " S " to score of " U);
                 } endif
           }  endfor
        }  endif
   } until ((Q is empty) or Size(Seen) > MaxPages);
```

Score

The score of link M in page P for query Q is computed as follows:

```
Score(M,P,Query) {
  if (Query == null) then return 0;
  if (K > 0 of the words in Query are substrings of M.anchor)   /* 1 */
    return K*50;
  if (any word in Query is a substring of M.URL)                /* 2 */
    return 40;
  U = set of different words in Query that occurs in P within five
      words of M (not counting HTML tags)
  V = set of different words in Query that occur in P;
  return 4*|U| + |V-U|
}
```

Normalize to lower case in matching.

For example, suppose Query is `walrus carpenter bread`.

- If M = `< A href = " file1.html " > WalrusAndCarpenter < /A >` then score=100.
- If M = `< A href = " walrus5.html " > Cute Poem < /A >` then score=40.

- If the word "walrus" appears in P both within 5 words of M and elsewhere in P, and the word "bread" occurs somewhere in P far away from M, then score = 5 (4 points for "walrus" plus 1 point for "bread"). The number of times these words appear in P is irrelevant.

Assumptions

You need only deal with HTML files. You need only deal with links that are indicated in the standard way in the HTML file: `< A href="URL" > Anchor < /A >.`

Crawlers

There is all kinds of code for crawlers on the Web, which you may use. As a starting point, I have written a minimal Web Crawler in Java. You can also look at the code described in Programming Spiders, Bots, and Aggregators in Java by Jeff Heaton, chapter 8. (Note: This is accessible online for free through an NYU account. You can also buy it in hard-copy, with a CD-ROM.)

Example

There is a simple, self-contained weblet here

Sample output for the query

```
java crawler -u http://cs.nyu.edu/courses/spring16/CSCI-GA.2580-001/MarineMammal/PolarBear.html -q ocean -docs ~/myaccount/ocean/ -m 7
```

Sample output for the query

```
java crawler -u http://cs.nyu.edu/courses/spring16/CSCI-GA.2580-001/MarineMammal/Whale.html -q whale whales species -docs ~/myaccount/whale/ -
```

CRAWLER COURTESY: VERY IMPORTANT

Crawlers are potentially dangerous. Therefore:

- Your crawler **MUST** observe the robot exclusion protocol. In the sample code linked above, this is implemented as the method `robotSafe`.
- Your crawler **MUST** always have a fixed upper bound on the total number of files to be downloaded. In developing, testing, and debugging, this number should be kept fairly SMALL.

Submission

Upload to the NYU Classes site your source code and a README file with instructions for compiling and running.