

Project 1: **Web Crawler**  
Search Engine Development  
100 points

A simple web crawler in written Java has been provided to you. You are to enhance the crawler in several ways:

1. The crawler should respect the robots exclusion protocol. Do this by looking for a robots.txt file for any new domain names encountered, and keep the list of excluded URLs in memory. Before a URL is to be added to the frontier, make sure it doesn't match any of the excluded URLs. For each crawl, you should only read robots.txt once for each domain.
2. The crawler should be started with any number of seed URLs. Instead of implementing this in the interface, just read in a **seeds.dat** file (in the current directory) which contains a list of seed URLs, one on each line. An example seeds.dat file would look like this:

```
http://taz.harding.edu/~fmccown/  
http://www.harding.edu/comp/  
http://en.wikipedia.org/
```

The file should be read immediately before the crawl begins.

3. The crawler should be limited in which websites it can crawl and how many pages. When your crawler begins to crawl, make it look for a file called **limits.dat** (in the current directory) which contains a list of host and domain names that should be crawled and the max number of pages that should be crawled from the domain. If a URL's domain name is not on the list, the URL should not be crawled. An example limits.dat file would look like this:

```
taz.harding.edu 10  
www.harding.edu 50  
en.wikipedia.org 100
```

According to the example, the URL <http://www.wikipedia.org/> would be rejected, but <http://en.wikipedia.org/test> would not. If the list is empty or the file is not found, there should be no limits placed on what domains can be crawled.

4. The crawler should properly normalize all encountered URLs according to the normalization rules discussed in class. In the crawler given to you, many of them are already done. The only required normalization you need to do is: convert the host and domain name to lowercase, strip off index.html and default.htm, and strip out session IDs that look like this: jsessionid=999A9EF028317A82AC83F0FDFE5938FF.
5. The crawler should save all downloaded files into a directory called **cache**, accessible off of the current directory. Each file should be named *hash*.html where *hash* is the MD5 hash (32 hex characters) of the page's normalized URL. An example filename is cf5eeb1371d85314c1f5983476df2d6a.html. By using a hash of the URL as a filename, subsequent crawls of the same URL will be saved in place of the previously crawled resource.

The following Java code will compute an MD5 hash on the given string:

```
import java.security.*;  
  
try {  
    MessageDigest md = MessageDigest.getInstance("MD5");  
    md.reset();  
    md.update("string to create hash on".getBytes());  
  
    byte[] arr = md.digest();  
    String hash = new java.math.BigInteger(1, arr).toString(16);  
}  
catch (NoSuchAlgorithmException e) {  
    e.printStackTrace();  
}
```

6. The URL of each successfully crawled page should be logged in an **urls.dat** file that resides in the cache directory. The file will record each normalized URL on a separate line. An example index.dat looks like this:

```
http://foo.org/  
http://foo.org/menu.html  
http://foo.org/log.html
```

When a new crawl is made, the URLs previously in the urls.dat file should remain. There should be no duplicate URLs in the file. You will probably find it easier to write to this file after the crawl has finished.

A good website to test your crawler is: <http://taz.harding.edu/~fmccown/fakesite/> Note that this site is only accessible within the Harding firewall. A robots.txt file is accessible from Taz which excludes a directory in fakesite.

### **10 Bonus Points:**

Allow the crawler to crawl any number of pages as long as they are within  $n$  hops of the root page. The user should be able to select the hop limit from a drop-down menu. By setting this limit, crawler traps will be avoided, and only the higher-quality pages will be found.

Submit your completed **WebCrawler.java** file to Easel before class on the day it is due.