COMP611 ALGORITHM DESIGN AND ANALYSIS

ASSIGNMENT 2
DUE DATE: 16 OCTOBER 2016

Instructions

(1) You may collaborate with (at most) one other student to complete this assignment.
(2) Each team shall submit a .zip file containing all the source files *and docs via AUTOnline*.
(3) On top of each file, please clearly indicate your names and student IDs
(4) You must clearly describe the main methods of your program (i.e., methods that implement main algorithms, etc.)

Marking Guide
The total marks of this assignment is 100, contributing 15% to final grade. Marks will be allocated based on the correctness, clarity and originality of your code. The details are:

- Question 1: 16 Marks
- Question 2: 20 Marks
- Question 3: 28 Marks
- Question 4: 28 Marks
- Question 5: 8 Marks

Total: 100 Marks

Introduction to Web Crawling. The purpose of this assignment is to develop an elementary web crawler that explores a part of the Internet in a breadth-first search manner, then use Page rank to develop a simple search engine. More precisely, a *spider* is a program that automatically ventures out on the Web and analyses documents. Their actions are based on the links between pages. You need to build a search engine by implementing a spider. For this assignment, you can get help from online resources (there are a lot of online tutorials explaining how to implement a web crawler).

The assignment is divided into the following steps:

Question 1. Create a program called `SpiderLeg.java` which parses HTML files from given URLs. For this task you can use `jsoup`, which is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods. You may download `jsoup` from the link:

`http://jsoup.org/`

And you may get a quick tutorial of `jsoup` from the link:

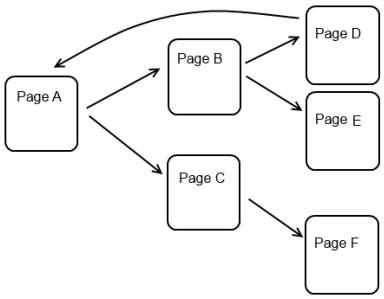
`http://jsoup.org/cookbook/`

Your tutor will take you through the use of `jsoup` in the lab. You may also find plenty of online materials that help you to develop an HTML parser using `jsoup`.

The file `SpiderLeg.java` should contain the following methods:

- `getTitle(String url)`: Print out the title of the web page `url`.
- `getHyperlink(String url)`: Print out all URLs in the web page `url`.
- `getImages(String url)`: Print out names of all image files in the web page `url`, as well as the height, width and alt attributes.
- `getMeta(String url)`: Print out the meta data, including meta description, and meta keywords of the web page `url`.

Question 2. Create a program called `Spider.java` that implements the spider algorithm for exploring the Web. The spider algorithm essentially implements a breadth-first search on the Web graph, where nodes are html pages, and directed edges are hyperlinks from one web pages to web pages. Generally speaking the algorithm works as follows:



INPUT:

- One or several *seed URLs*,
- Other relevant information, e.g. a keyword to be searched
- an integer d indicating the depth of exploration (so you don't work with the whole Internet)

OUTPUT:

- certain data that are involved in all pages that are distance d away from the seed url(s)

Data structures:

(1) A list of unvisited URLs - seed this with one or more starting pages
(2) A set of visited URLs - so you don't go around in circles (you should use a file for storing this list as it gets big)
(3) Some rules for URLs you're not interested - so you don't index the whole Internet

Pseudocode:

```
while(list of unvisited URLs is not empty) {
    take URL from list
    use SpiderLeg to fetch content
    if content is HTML {
        use SpiderLeg to parse out URLs from links
        for each URL {
            if it matches the rules and not already visited or in the unvisited list
                add it to the unvisited list
        }
    }
}
```

Also to keep the number of web pages you explore manageable, you may associate with every URL in the unvisited list an integer value indicating its distance from the seed pages. If the distance is larger than the depth of exploration d, then you stop exploring further in the Internet.

Your task for this question is to implement a `crawl` method that crawl the Web from a set of given seed URLs and print out all the URL that the spider visit.

Question 3. This tasks asks you to build a search tool on the Internet. You should build a simple user interface (command line or GUI) which allows the user to specify *seed URLs* as well as a *key word* to be searched. Then the search tool should use a spider to explore the Web, searching for web pages whose meta keywords contain the specified key word. The results should be a list of URLs. The program should store these URLs in a file, as well as display them to the screen.

Question 4. This tasks asks you to extend your search tool to a simple search engine by incorporating the Page ranks of resulting web pages. You should implement the power iteration algorithm for computing Page ranks of all visited web pages. Then in the search tool, whenever the user

performs a search of a key word, the program should display the resulting URLs in decreasing order of Page ranks.

Question 5. Write a document (.pdf) of your search tool including all functionalities. This should serve as a manual that allows the user (and the marker) to understand how your search tool works.