ICS 211 Extra Credit Assignment - A Java Webcrawler

March 1, 2021

This is an extra credit assignment worth 5% of your grade. This will be an in-depth intro to several important concepts that you will explore further in later classes. The first exercise will be worth 1% and will cover regular expressions, an important topic in several computer science areas. The next exercise covers extracting links from an HTML document and is worth 2%. The final portion is worth 2%. In this exercise you will take the two previous exercises and use them to implement a webcrawler that will take data from a webpage, then search through it using the graph traversal algorithms.

Exercise 1

In this exercise, you will use regular expressions to filter input and find web addresses. A regular expression is a search pattern used to find and format strings. The **Pattern** class contains all the information you will need to use a regular expression.

Once you have read and learned about regular expressions, it's time to put them to use. Create a public class called **WebCrawler**. This class will also need a main method. The program will accept two command line arguments: the first will be a website in the form of a string, the second will be an integer that will represent how deep the crawler will search in part 3. Use regular expressions to verify that the first argument represents a valid web address. A web address starts with either http or https, then "://", then some string, then a period, and finally an ending of com, edu, mil, org, or gov (Note: there are also several country codes, but you can assume U.S. web addresses for this assignment). These endings are known as top level domains, and each has a specific use. Feel free to research further if you are interested. If the first argument does not fit the pattern of a web address, the program should exit gracefully, giving the user a message and showing how to properly use the program. Example

Listing 1: WebCrawler Error Example

```
Error: invalid web address
Usage:
WebCrawler [valid web address] [depth to be searched]
```

Exercise 2

In order to do this exercise, we must first discuss HTML documents. HTML, or hypertext markup language, is the frame for a webpage.

Listing 2: Example of an HTML File

```html
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
        }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>
<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

There is quite a bit of information contained in an HTML document, and it would be beneficial to study this. For this assignment, we are only concerned with line 40. This line is a hyperlink that says "More Information..." that when clicked will take the user to https://www.iana.org/domains/example. The links we want to extract from a document will begin with **href="** and end with a **"**. Use this pattern to extract links from these documents.

Now we will discuss how we get information on the web. HTTP is the hypertext transfer protocol, and is the protocol clients and servers use to host and get information. For the purpose of this assignment, you only need to know that a GET request is used to get a webpage. This topic is once again very dense, so to assist you, I've posted the snippet you will be using for this portion below.

Listing 3: Additional Packages You Will Need

```java
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
```

Listing 4: Java GET request snippet

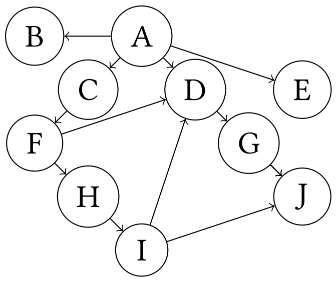
```java
    HttpClient client = HttpClient.newHttpClient();
    HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(args[0]))
            .GET()
            .build();

    HttpResponse<String> response = client.send(request,
            HttpResponse.BodyHandlers.ofString());
```

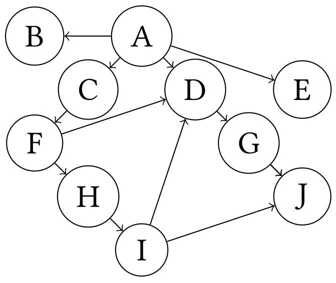
Once you have the response, you can parse it using the regular expression you created to extract links from HTML documents. If you choose not to continue to part III, print the links you extracted to get credit.

Exercise 3

The final exercise will be to implement two graph traversal algorithms in order to traverse the links you've gathered. We can represent the web as a graph, with web pages as nodes and the links between them as edges. There are two algorithms you will use: A breadth-first search and a depth-first search. Consider the graph below:



Assume the graph is traversed from the top left-most nodes to bottom right In a breadth-first search, assuming we began with node A, we would first examine the nodes adjacent to A, B, C, D, and E. We would then move to B, and examine it's adjacent nodes. Since B has none, we would move to C, and explore it's adjacent node, F, then to D, then to E, etc. A breadth first search would traverse in the order ABCDEFGHIJ.



A depth first search on the other hand examines an adjacent node and immediately moves to it. Once a given node has had all of it's adjacent nodes explored, we move back to the previous node and explore the remainder of it's nodes. For example, We would begin at A, then move to B. Since B has no adjacent nodes, we move back to A and explore C, then F, etc. The traversal would look like ABCFDGJHIE.

Now that you've had a soft introduction to these graph traversal algorithms, implement them in your code. You should use the second command line argument as the depth to traverse, for instance if the argument is two, you should explore links two away from the initial web address. Use two different data structures to store the links you've found, one for BFS and one for DFS. Be sure to avoid traversing links you've already seen. You can reuse the code I provided in Exercise 2 to get the HTTP response. Print to the console when you begin a traversal, and have it print the link and depth of the link to the command line each time it accesses a new link. Please contact the TA with any further questions. Good Luck!