Homework 6: Web Compressor

Due Date

Friday, May 13 @ 2 hours after 11:59pm (2am Saturday morning)

Topics

- Graph Representation
- BFS

Resources

WebGraph: https://en.wikipedia.org/wiki/Webgraph (https://en.wikipedia.org/wiki/Webgraph)
C++ Reference: http://www.cplusplus.com/ (http://www.cplusplus.com/)
Python Documentation: https://docs.python.org/3/ (https://docs.python.org/3/)
Javadocs: https://docs.oracle.com/javase/7/docs/api/ (https://docs.oracle.com/javase/7/docs/api/)
Google is Your Friend: https://www.google.com/ (https://www.google.com/)

Graph Representations

Implement the following functionality for graphs. These functions will work on any arbitrary graph, though testing will be performed on a graph of the web and the phrasing of the assignment will be oriented towards web graphs. The data is a small sub-graph of the full web to allow faster development and testing, though the graph is still fairly large.

Testing Graph: http://ubcse.com/data/webGraph.zip (http://ubcse.com/data/webGraph.zip)

Format: The data is contained in two tab separated files. The nodes file gives the mapping of url to id ("buffalo.edu\t6136198") and the edges file gives every edge by the node id's in the order of source node then destination node ("15328636\t6136198"). This edge means that site "15328636" contains a link to site "6136198".

Data sources: The data was obtained from http://webdatacommons.org/hyperlinkgraph/2012-08/download.html (http://webdatacommons.org/hyperlinkgraph/2012-08/download.html) which was extracted from Common Crawl (http://commoncrawl.org/ (http://commoncrawl.org/)). To reduce the size of the graphs, we'll use the pay-level-domain graph that has been filtered to only include the most popular sites according to Alexa (http://www.alexa.com/) (http://s3.amazonaws.com/alexa-static/top-1m.csv.zip (http://s3.amazonaws.com/alexa-static/top-1m.csv.zip)). The data has been filtered down to 101,797 sites connected by 8,303,699 hyperlinks. Note that the graph is from 2012 so many newer sites will not be present.

Sample test cases are provided here (homeworkCode/hw6TestCases.zip). These are the test cases that will be used for grading and can be used to check the functionality of your code.

For a resource of graphs in a format that is compatible with your submission you can see the SNAP data sets from Stanford (http://snap.stanford.edu/data/ (http://snap.stanford.edu/data/)).

Prune the Web (1 point)

```
<executable name> –prune nodeFilename edgeFilename outputFilename k
```

Return a graph containing only sites with k or more sites linking to it. The returned graph can contain sites with fewer than k incoming edges as long as they had k or more links in the original graph.

Create a file with the name contained in the outputFilename parameter with the edges of the new graph in the same format as the input edges file.

Shortest Navigation Path (1 point)

```
<executable name> –path nodeFilename edgeFilename outputFilename startURL destinationURL
```

Write the edges into the output file of the shortest path from startURL to destinationURL. Ties in distance between multiple paths can be broken arbitrarily. For testing we will check if the length of the path matches the shortest distance and that the path is valid.

The format of the output should be a list of edges by domain name (not node id) in the order of the path.

Note that we are working with pay level domains so this is not the minimum number of clicks to get from one site to another, but it is the shortest domain name path.

The Cover-Up (1 point)

```
<executable name> –cover nodeFilename edgeFilename outputFilename k url
```

Given a domain name, remove it from existence! In addition to removing the given url from the web graph, the program must also remove any site linking to the domain with distance k or less.

Create a file named outputFilename with the edges of the new graph in the same format as the input edges file.

Efficiency (1 point)

The cover-up must run in under 1 minute.

Running WebGraph

There will be no provided code for the primary goals of this assignment and your submission must follow the conventions for your language of choice. The executables will be ran and tested for the properties above. Official compiler/interpreter/vm versions will match those installed on timberlake.

- C++
  - Submit files named WebGraph.cpp and WebGraph.h
  - Compile: "g++ -g -std=c++11 WebGraph.cpp"
  - Run: "./a.out <params>".

- Java
  - Submit a file named WebGraph.java
  - Use the default package (do not specify a package). Compilation will be called in the same directory as WebGraph.java.
  - Compile: "javac WebGraph.java"
  - Run: "java WebGraph <params>"
- Python
  - Submit a file named WebGraph.py
  - Run: "python WebGraph.py <params>"

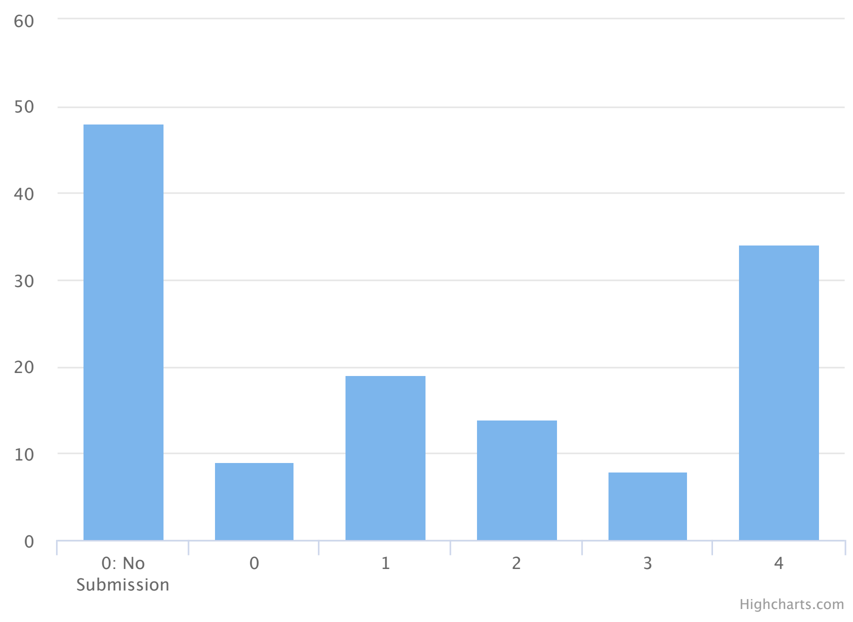
Submission

Submit: WebGraph.cpp/WebGraph.h || WebGraph.java || WebGraph.py
Submission must be made on the submission website: https://www-student.cse.buffalo.edu/planetexpress/ (https://www-student.cse.buffalo.edu/planetexpress/)

Solution

WebGraph.java (homeworkSolutions/WebGraph.java)

Results


0: No Submission
Highcharts.com