Data Structures and Algorithms, CS146, Fall 2018
Programming Assignment II
Due at 11:59pm, on November 16 (Friday), 2018
(No extension and no excuse for late submission!)

Note:

This programming assignment is for individual work only. You are not allowed to use someone's codes or copy from internet, except for Web Crawler codes. Code plagiarism checker tools (Jplag, MOSS, etc.) will be used to check the similarity of codes. Please check on the University Policies in the syllabus.

Read this instruction carefully before your design and implementation of your work.

Programming Motivation

The best way to learn data structures and algorithms is to write codes. Writing computer program code to solve problem using data structures and algorithms is also a required skill for a software engineer working for IT and software industries. Hands-on coding challenges will strengthen your programming skill and experience. In fact, every job interview event/opportunity that you will be encountering in the near future always test your coding kills to see if you can solve problems. If you are majoring in computer science or software engineering major, you **CANNOT** avoid coding, unless you study CS and SE just to kill time for fun and you are not pursuing for your bright IT career future.

Therefore, enjoy your time in coding and have a lot of fun!!!!

Problem Statements

Based on the web crawler and data structure for the Simulation of Google Search Engine you developed from the PA1 (if you didn't or you built a bad one, it is the time for you to retry and develop a nicer one), you are a Software Engineer at Google and are asked to conduct the following Google's Search Engine Internal Process:

1. Based on scores of 30 URLs that you retrieved from your web crawler, use Quicksort algorithm to sort the scores for PageRank instead of using Heapsort from PA1. (You MUST print a list of 30 URLs including Index, total score, PageRank, and URL. If not, your will lose 50 points.)
2. Based on scores of 30 URLs, use Binary Search Tree to manipulate the data. The following section list the programming requirements for BST.
3. To speed up the search result, Google search engine dynamically collects a list of top N popular keywords (N=10 for instance in PA1) and use Bucket Sort to sort their company's names of URLs in alphabetical order (example: http://www.abcde.com). You can store company's name starting with A in bucket "A", starting with B in bucket "B", and so on.

Programming Requirements

1. The Google Search Engine Internal Process MUST be written in Java and it is required to use pseudo codes in the textbook. (Please be noted that you MUST use the pseudo codes provided in textbook to write your Java codes. Any other codes will be treated as failing requirements and **will receive 0 points**.)
2. You need to use a Web Crawler to enter keywords and then each keyword will receive 30 URLs for further Google Internal Process. (You will receive 0 if you hardcode 30 URLs in your program/code.)

3. You must follow the four PageRank factors from your PA1 to calculate score for PageRank.
4. Your simulation application **MUST** at least contain the following functions for **URL Binary Search Tress (BST):**
    a) Build up a Process BST. (**MUST follow BST properties** specified in textbook and ppt slides. Your own tree structure will not be accepted.)
    b) Users can search a specific PageRank and show the specific URL (User want to know the score of a specific website.).
    c) Users can insert a URL to the BST based on its total score and show the result.
    d) User can delete a URL from the BST and show the result.
    e) Users can make a sorted list of URLs according to score from the BST and show the result.
    f) To show the result, you MUST print a list of URLs including Index, total score, PageRank, and URL. If not, your will lose 50 points.
5. Each java file/class/subroutine/function call must contain a header comment in the beginning of it. (You will lose 20 points for not putting comments in your program.)
6. You **MUST** write a **professional** MS Word or pdf report with the following items included in the report:
    a) Cover page.
    b) Explain/Illustrate your design and implementation details. (in terms of Data Structure, user interface, etc. -20% if not provide or no detailed explanation.)
    c) A list of classes/subroutines/function calls and explanations of each purpose. (-20% if not provided)
    d) Provide a lot of self-testing screen shots of each functional requirement including inputs and outputs for each of functional item listed in Problem Statement. (You will receive 0 point for your PAII, if no self-testing screenshots provided.)
    e) Each result display of your program outputs MUST include Index, Score, PageRank, and URL. (**If testing outputs only show scores without URL will receive 0 points**)
    f) The procedure (step by step) of how to unzip your files, install your application, and run/test your codes. (-30% if not provided)
    g) Problems encountered during the implementation. (-10% if not provide or too simple)
    h) Lessons Learned (-10% if not provided or too simple)

Submission Requirements

1. The deadline to submit/upload your report and source codes to Canvas is 11:59pm on Friday, 11/16/2018. (**No Late submission!** Please do not wait until the last minute to upload and submit. If fail to submit, **no excuse for this time!**)
2. Zip all your files including your report, all *.java files, and a Java executable file (a jar file, so that I can run your codes) into one zipped file with file name: **Your_First_Name_Last_Name-PA2.zip**, Any file with incorrect file name will not be accepted and 0 point will be given.
3. **Jar file MUST be runnable (To save my time to run your program and do grading, -50% if not runnable**.) Runnable: at least, allow to run your jar file at command line or console.
4. Grading is based on **Full functioning**, **completeness of requirements, clarity of your codes,** and **professional report**. (**No Report submission -100%**)