CS6675/ CS4675

# Homework Assignment 1
(Programming Category)

Student Name:_______________________

Student Session: cs6675 or CS4675 (circle one)

You are given the choice of two programming problems about Web crawl or Web Search in the first homework assignment. The first programming problem consists of multiple options. You only need to choose one option from one of the problems as your first homework. Feel free to choose any of your favorite programming language Java, C, Perl, Python.

**Due Date**: Midnight on Friday of Feb 4 with graceful (no penalty) extension till 9am on Feb 5 (Sat). **No late submission will be accepted.**

## Problem 1. Hand-on Experience with a Web Crawler

This problem has a number of options. You are expected to perform one of the following two options:

**Option 1.1: Experience with an Open Source Crawlers**

(1) Download and install a Web Crawler, such as the open source crawler from Apache Lucene search engine (http://lucene.apache.org/core/), or PeerCrawl (http://www.cc.gatech.edu/projects/disl/PeerCrawl/). You may also find other alternatives from https://en.wikipedia.org/wiki/Web_crawler
(2) Select a seed URL to initialize your Web crawler, such as cc.gatech.edu, and you are expected to crawl at least 1000 URLs.
(3) Show the design of your Web archive to store your crawled web pages, the keywords (subjects) you have extracted.
(4) Plot the crawl speed in terms of the number of keywords you have extracted and the number of URLs you have extracted and the number of URLs you are able to crawl.
(5) Discuss your experience and lessons learned. Predict how long your crawler may need to work in order to crawl 10 millions of pages and 1 billion of pages.

Deliverable:

(a) Source code and executable with readme.
(b) Screen Shots of your Web Crawler's Command lines or GUIs
(c) Crawl Statistics (Crawl speed→ #pages/minute, ratio of #URL crawled / #URL to be crawled, etc.) in excel plots or tabular format
(d) Discuss your experience and lessons learned.

**Option 1.2: Write a Web Crawler of your own.**

Feel free to use any open source crawler as the code base. Write your focused crawler, such as crawling only CoC website or crawling only healthcare web pages. You are expected to crawl at least 1000 pages.

Here are some steps you may follow in a similar way as those in Option 1.1:

(1) Select a seed URL to initialize your Web crawler, such as cc.gatech.edu, and you are expected to crawl at least 1000 URLs.
(2) Show the design of your Web archive to store your crawled web pages, the keywords (subjects) you have extracted.
(3) Plot the crawl speed in terms of the number of keywords you have extracted and the number of URLs you have extracted and the number of URLs you are able to crawl.
(4) Discuss your experience and lessons learned. Predict how long your crawler may need to work in order to crawl 10 millions of pages and 1 billion of pages.

Deliverable:

(a) Source code and executable with readme.
(b) Discuss the design of your crawler: Pros and cons.
(c) ScreenShots of your Web Crawler's Command lines or GUIs
(d) Crawl Statistics (Crawl speed→ #pages/minute, ratio of #URL crawled / #URL to be crawled, etc.) in excel plots or tabular format
(e) Discuss your experience and lessons learned.

## Problem 2. Hand-on Experience with Open Source Search Engine

In this programming assignment, you are asked to gain **hand-on experience with an open source search engine system,** such as Apache Lucene search engine (http://lucene.apache.org/core/) or any of your favorite search engine packages.

Consider Lucene as an example. You can read tutorial at http://www.lucenetutorial.com/lucene-in-5-minutes.html.

You are expected to create a toy search engine with a minimum of 100~1000 documents (can be web pages from a public web page repository or crawled by Lucene crawler, for example).

**(1)** Complete the setup of your toy search engine system with initial 100 documents, show your index with 5 example keywords (index terms).
(2) Add 1 web document and show that this document will be indexed and searched using your toy search engine. Also show the update of your index.
(3) add 100 documents and show that all 100 documents are indexed and searched by your toy search engine. Also show the update of your index.
(4) Perform measurement on adding K documents to your toy search engine. For example, measuring performance of adding K documents and display the time cost on y-axis with x-axis varing K from 100, 200, 300
(5) Discuss your observations and elaborate your discussion, including critique on the toy search engine (pros and cons) and your suggestions for improvement.

Deliverable:

(a) Source code and executable with readme.
(b) Discuss the design of your crawler: Pros and cons.
(c) ScreenShots of your toy search engine command lines or GUI
(d) Index and search performance in excel or tabular format
(e) Example index entries.
(f) Discuss your experience and lessons learned.