University of Windsor

School of Computer Science

COMP-8547 Advanced Computing Concepts – Winter 2020

Lab Assignment 4

Deadline: **All sections July 17, 2021 at 11:59pm**

Rules: This assignment must be submitted and will be evaluated individually. You are allowed to discuss the problems and exchange some ideas within a group or with other students. However, the source code for the programs should not be identical among those students.

You are required to use Java SE 8 and Eclipse to write your programs, and the data structures and implementations as discussed in class.

Objectives: The aim of this assignment is, and text processing algorithms for pattern search and string matching. It will also help understand regular expressions and regular languages, and the basic principles of a tool for parsing and translation. Students will obtain hands on experience in writing, implementing and using these algorithms on applications to real problems in searching text and finding patterns in Web pages using Java SE 8 and Eclipse.

Tasks:

1. Use classes BruteForceMatch, BoyerMoore and KMP provided in the source code.
   a. Download file Hard disk.txt from the Resources.
   b. Find all occurrences of patterns “hard”, “disk”, “hard disk”, “hard drive”, “hard dist” and “xltpru”, and show the offsets.
   c. Repeat (b) 100 times and record the average CPU time for each case.
   d. Compare the CPU times with the running times of the three algorithms (discussed in class) and comment on asymptotic running time of the corresponding algorithms.
2. Download file Protein.txt from the Resources. Using class TST provided in the source code:
   a. Write a program that reads file “Protein.txt” and creates a trie using TST. Use StringTokenizer, Jsoup or a similar API to extract the words from the file.
   b. Do several searches of keys “protein”, “complex”, “PPI”, “prediction”, and others, and show the occurrences of these words in file Protein.txt
3. HTMLtoText converter: Write a program that takes the 100 given Web pages (W3C Web Pages.zip), and using Jsoup, converts all files into text. The text files should be saved as individual files for use in the next tasks of this assignment. Keep good OO design practice by creating a method processes one file. That method will then be called 100 times.
4. Pattern finder: Using Java Regex, find phone numbers and email addresses in the 100 **text files**.
5. URL finder: Using Java Regex, write a program that finds Web links (URLs) in a Web file.
   Test your program with the 100 **HTML files** to find the following: a.
   Links with domain w3.org
   b. Links that contain folders: e.g., www.w3.org/TR/owl-features/
   c. Links that contain references in a Web page and may contain folders; for example: www.w3.org/TR/owl-features/#DefiningSimpleClasses

d. Links with domain .net, .com, .org

Submission:

1. You must submit: (i) a report (in PDF or word), which contains the answers to all questions, (ii) all java files and classes needed to run your programs. Your programs should run in Java SE 8 in the Eclipse IDE. You should upload all this files to the Blackboard, as a **single** zip file.
2. In your report, you must provide written answers to all questions, including the programs' outputs, as required. Marks will be deducted if comments/explanations are missing.
3. Add the following note at the beginning of your report:
   "I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work." + Name + SID
4. Any submission after the deadline will receive a penalty of 10% for the first 24 hrs, and so on, for up to three days. After three days, the mark will be *zero*.
5. Unlimited resubmissions are allowed. But keep in mind that we will consider the last submission. That means that if you resubmit after the deadline, a penalty will be applied, even if you submitted an earlier version in time.

Application: Web search engine (Optional)

1. Inverted index: Write a program that using the Web pages given in the resources, it constructs an inverted index, as explained in class. The words (keys) are stored in a trie. The value of each word is the index to a list of occurrences of that word in each Web page. You can keep the lists of occurrences in a two-dimensional array as explained in class.
2. Web search engine: Write a program that given a keyword (a single word), it shows a list of Web pages in a ranking from high to low (sorted in decreasing order of occurrence of that word).
3. String matching: Write another program that implements the search index from another perspective. Given a keyword and a web page, the program should use string matching to find how many occurrences of that keyword are in a particular web page. The program should do this for all Web pages, and rank the pages based on then number of occurrences. Note that the programs for string matching we've seen in class find the first occurrence only. You will have to modify the matching algorithm to find all occurrences (or at least to count them).