CS 6350: Assignment 0

Due on Wednesday, August 30 2017 11:59 PM

Fall 2017

Contents

Instructions

- This assignment will help you refresh your Unix and Java programming skills.
- It will also help you get familiar with the UTD Unix machines and programming environment that we will use later for Big Data programming.
- You are allowed to work in pairs i.e. group of 2 students or individually.
- You should use a cover sheet, which can be downloaded at:
  http://www.utdallas.edu/ axn112530/cs6350/CS6350_CoverPage.docx
- Your program will need to work on the UTD Unix machines. You can logon to the machines by using a SSH client, such as Bitvise, PuTTy, or Mac iTerm. More details about UTD Unix machines are available in problem 1.
- This assignment is due on **Wednesday, August 30 2017 11:59 PM**. There will be no extensions of the due date. Please submit your assignment via eLearning. Other methods of submission - e.g. email to the instructor or TA will not be accepted.
- You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After that, there will be a penalty of 10% for each late day. The submission for this assignment will be closed 2 days after the due date.
- Please ask all questions on Piazza, and not through email to the instructor or TA.
- No individual exceptions are allowed. The above rules apply to every student of the class.

Problem 1

Unix Programming

In this part, you will first familiarize yourself with the Unix programming environment at UTD and write a script to download a large text file and compute the frequency of words in it. The steps for this part are as below:

1. Logging to UTD Unix cluster

You will need to SSH into UTD Unix machines. You can use one of SSH clients, such as Bitvise, PuTTy, or Mac iTerm. The address of the machines are:

- csgrads1.utdallas.edu
- cslinux.utdallas.edu
- cslinux.utdallas.edu

It is preferred that you use csgrads1.utdallas.edu, but if that doesn't work, you can try the others. As a reminder, the command to SSH is something like:
ssh YourNetID@csgrads1.utdallas.edu
For example,
ssh abc123456@csgrads1.utdallas.edu

2. Word Count Problem

Counting the frequency of words in a large text file is known as the *word count* problem and is widely studied in text mining and data analytics. In this part of the assignment, you will write a Unix script that will first download a large text file and then find the frequency of each word in the file and then write the output to a file in decreasing order of word frequencies. You will do all this using **only Unix commands** that can run on the UTD machines.

Below is the process broken down into steps:

1. **Download text file** - The first step is to download a large text file into your file system. You will choose a text file from the Gutenberg collection - https://www.gutenberg.org/. For example, if you want to download the book "A Tale of Two Cities" by Charles Dickens, you would download it from the direct link https://www.gutenberg.org/files/98/98-0.txt. You can use a command such as **curl** or **wget** in Unix to download the files.
2. **Tokenize and Count Word Frequencies** - The next step is to break down the text file into tokens (words), and then evaluate the frequency or count of each word. You will find Unix commands **tr**, **wc**, **sort**, and **uniq** helpful. You are free to use other commands also.
3. **Write output to a file** - The final step is to sort the output in decreasing order of word frequency i.e. highest count words first and write the output to a file named **"YourNetIDPart1.txt"**. For example, if your netid is abc123456, you will name your file **"abc123456Part1.txt"**

3. What To Submit

You will submit the following for this part:

- Your Unix shell script file
- README file indicating which text file you chose and how to run your command file

Problem 2

Java Programming

In this part, you will program the same word count problem that you did in part 1 using Java. You are free to use advanced Java data structures such as HashMap - https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html to store the intermediate data, such as key-value pairs.

1. Develop Java Code

The first step is to write Java code to download the same file that you used in part 1. You can use Java classes, such as Java NIO - https://en.wikipedia.org/wiki/New_I/O_(Java) or
Java URL class - https://docs.oracle.com/javase/tutorial/networking/urls/readingURL.html to download the text file.

The second step is to extract word frequencies from the text file. You can use advanced Java data structures, such as HashMap https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html to store the intermediate data, such as key-value pairs. Remember that you have to sort the data in decreasing order of word frequencies.

The third step is to store the output to a file on the Unix filesystem that contains word counts in decreasing order of word frequency i.e. highest count words first to a file named **"YourNetIDPart2.txt"**. For example, if your netid is abc123456, you will name your file **"abc123456Part2.txt"**.

You can use any Java platform or IDE to develop your code locally and compile it into a jar file, as long as your code can run on the UTD Unix machines described in part 1 of this assignment.

Note: Some of you may be familiar with the MapReduce approach to solve this problem. However, you are NOT to use that approach, but to solve it using normal Java programming and data structures.

2. What To Submit

You will submit the following for this part:

- Your Java code file
- Your compiled jar file
- README file indicating which text file you chose and how to run your jar file.