CompSci 516: Homework 2

Spring 2022

Total Points : 100

- Part-I Posted on : Thursday, February 10, 2022
- Part-II Posted on : TBD
- Due on (both parts): Tuesday, March 1, 2022 12:00 PM (noon)
- Note: You can start working on it right away.

Course Policy Reminder

Please remember that you need to **strictly** follow the following rules while you are working on the homework:

1. ☐ You can NOT work in a group to solve the problems.
2. ☐ You CAN install a software together with other students and test whether toy examples run (NOT any homework question).
3. ☐ You CAN discuss possible approaches and troubleshoot once you have put enough efforts yourself, acknowledging all such discussions with students' names in your submission. You must OWN your solution and be able to defend it if asked.
4. ☐ You can NOT show your solution to any question to any other student.
5. ☐ You can NOT see the solution to any question from any other student.
6. ☐ You can NOT find the solution to any question from the Internet or any other sources. Note that you can and need to learn programming languages, frameworks, and libraries using online tutorials, and then need to write your own code.
7. ☐ If you are unsure about what is allowed, you need to send the instructor an email and ask about it.
8. ☐ Of course, feel free to post questions on Ed (as many times as you need) if anything is confusing or if you need help! Anonymous questions are fine but do not copy your solutions. If you have specific questions about your solution, you can send private questions on Ed.

1 Overview

Our goal in this assignment is to introduce you to Apache Spark, a distributed compute engine that could process large data (http://spark.apache.org/).
In this assignment, you will write two Spark applications in Scala programming language, test them on your local machine with small input files, then run them on a cluster of AWS EC2 instances with large input files. If you are new to Scala, we will expect you to pick up the language on your own as you work on this assignment (each application is about 50 lines of code).

2 Part-1 : Spark on Local Machine

In this section, you will write two Spark applications in Scala and test them on your local machine with small input files.

2.1 Install Spark

Install Spark on your local machine following the tutorial provided on Sakai:
`CompSci_516_HW2_spark_tutorial-local.pdf`.

2.2 Download Template File

Download `hw2_code.tgz` from Sakai. To untar the file, open up a terminal then type:

```
$ tar -xvzf hw2_code.tgz
```

And you should have:

```
$ cd hw2_code$
$ find .
.
./sample_input
./sample_input/titles-sorted.txt
./sample_input/links-simple-sorted.txt
./output.txt
./build.sbt
./src
./src/main
./src/main/scala
./src/main/scala/PageRank.scala
./src/main/scala/NoInOutLink.scala
```

`hw2_code/` contains template code and sample input files. You will have to fill in the template code, and you could use the given sample input files to test your programs on your local machine. `TODO` tags and `???` tags are placed to help you complete the logic, but you can implement your own code without following those tags. `output.txt` is what you need in Part-2.

2.3 Input

1. `links-simple-sorted.txt:`
   `4: 1 2` means there are two links from webpage 4. One points to webpage 1, and another one points to 2. So this is two out-links for webpage 4, one in-link for webpage 1, and one in-link for webpage 2. The numbers are only the indexes for the actual webpages. See the explanation for `titles-sorted.txt`.

2. `titles-sorted.txt:`
   `C` means the actual title for webpage 3. This input file specifies a mapping from the index to the title. You should infer the index for each title from its line number (starting from 1) in the file.

These are synthetic data, so the link title doesn't mean a real webpage address.

2.4 No In/Out Link

Write a Scala program, `NoInOutLink.scala`, that:

1. outputs the first 10 pages with no outlinks (ascending order in index).
2. outputs the first 10 pages with no inlinks (ascending order in index).

With the given sample input files, your program should print out:

```
[ NO OUTLINKS ]
(1,A)

[ NO INLINKS ]
(7,G)
(8,H)
(9,I)
(10,J)
(11,K)
```

Note that getting the above output does not necessarily mean your solution is correct. Validating the correctness of your implementation is your own responsibility.

2.5 PageRank

In this section, you will implement the PageRank algorithm using Spark API and output the 10 pages with the highest pagerank. Here is the details of the PageRank algorithm.

$$PR_0(x) = 100/N$$
$$PR_i(x) = \frac{(1-d)}{N} \times 100 + d \sum_{y \to x} \frac{PR_{i-1}(y)}{out(y)}$$

- $PR_i(x)$ – the pagerank of node `x` at i^{th} iteration.
- `d` – damping factor (we will set it to 0.85).
- `N` – the total number of pages in the system.

- `out(y)` – the number of outlinks of node (webpage) `y`.

Run your program for 10 iterations ($PR_{10}(x)$ for all nodes).

For more information, please read https://en.wikipedia.org/wiki/PageRank.

With the given sample input files (`hw2_code/sample_input/`), the ranks of each page should look similar to Figure 1, and the ranks should add up to 100. With the given sample input

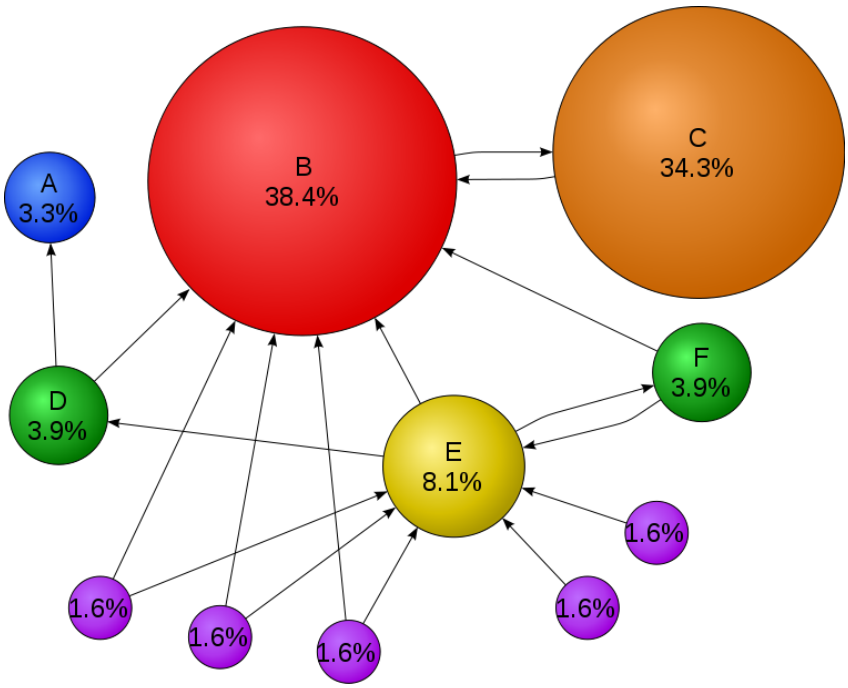

Figure 1: Image from `https://en.wikipedia.org/wiki/PageRank`.

files, $PR_{10}(x)$ for all nodes should be

```
[ PageRanks ]
(3,(C,36.72612315790154))
(2,(B,36.00158480709947))
(5,(E,8.08854359207413))
(4,(D,3.9104616734286384))
(6,(F,3.9104616734286384))
(1,(A,3.278179407194857))
(7,(G,1.616929137774546))
(8,(H,1.616929137774546))
(9,(I,1.616929137774546))
(10,(J,1.616929137774546))
(11,(K,1.616929137774546))
```

Note that you should only print out top 10 of them rather than all the 11 ranks. And the order of pages with the same rank doesn't matter.

Part-2 of the homework is on AWS EC2 and also includes the submission instructions.