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.

2 Part-1 : Spark on Local Machine (50 points)

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: `spark_tutorial-local.pdf`.

2.2 Template File

`hw2_code.tgz`. This file contains the 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 loacl machine. In order to untar the file, open up a terminal then type:

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

You should find files as:

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

For files in `sample_input`:

1. links-simple-sorted.txt:

   2: 3 → This means in the webpage of link 2, there is a link that points to link 3. So this is an out-link for 2 and an in-link for 3. The number here is just an index or key for the actual link. See the explanation for titles-sorted.txt

2. titles-sorted.txt:

   A → This the title for the link 1. So this file actually specifies a map from the index to the title. You should infer the index for each title from its position in the file. The rule is, if it appears in the i_th row (starting from 1), then the index for this title is i.

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

2.3 No In/Out Link

Write a Scala program, `NoInOutLink.scala`, such 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.

2.4 PageRank

In this part, 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` – a damping faactor (we will set it to 0.85).
- `N` – the total number of pages in the system.
- `out(y)` – the number of outlinks of node `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.

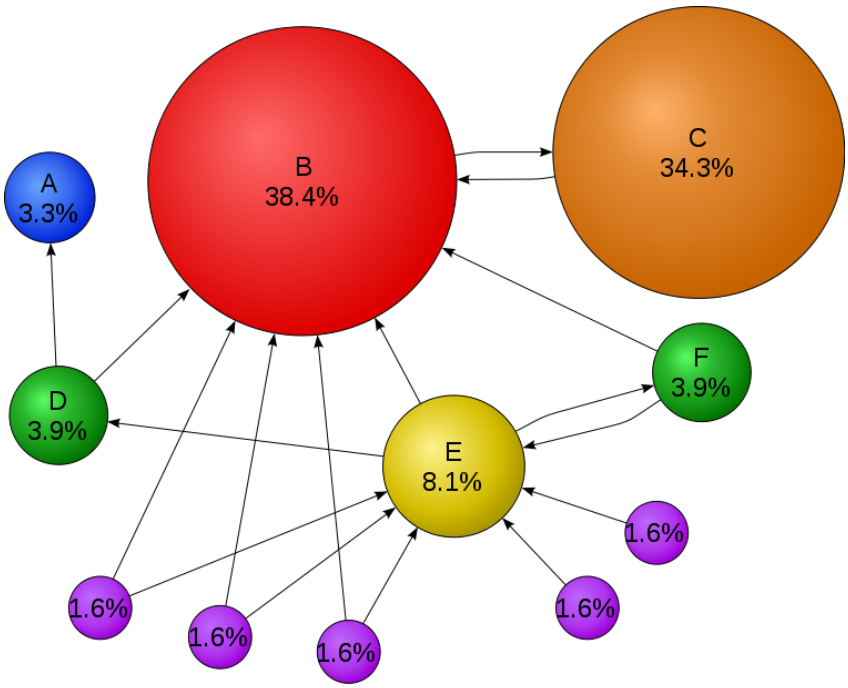

Figure 1: Image from `https://en.wikipedia.org/wiki/PageRank`.

Thus, with the given sample input files, your program should print out the result similar to below:

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

If you keep running, the number will converge and you will get the result similar to this:

```
(2,B,38.4)
(3,C,34.3)
(5,E,8.1)
(4,D,3.9)
(6,F,3.9)
(1,A,3.3)
(7,G,1.6)
(8,H,1.6)
(9,I,1.6)
(10,J,1.6)
(11,K,1.6)
```

3 Part-2 : Spark on AWS EC2 (50 points)

In this section, you will instantiate a cluster of AWS EC2 instances and run your Spark applications on the cluster. In the end, you will need to turn in:

- `NoInOutLink.scala` and `PageRank.scala`
- `output.txt` that contains the output of your applications
- `report.pdf` that explains how the cluster execute your applications using snapshots taken from Spark's web UI.

3.1 Download Input Files

Download the following input files from Sakai:

- `links-simple-sorted.zip`
- `titles-sorted.zip`

There are of the same format as ones you use in Part-1 but with larger volume.

3.2 Launch Spark Cluster

Instantiate a cluster of EC2 instances that will run your Spark applications.

- 1 master + 10 workers
- instance-type: t2.large (2 CPU, 8GB RAM)

There is a step-by-step tutorial: `spark_tutorial-aws.pdf`.

Important: Please terminate the cluster after you are done using it. You are paying for each EC2 instance you use every hour.

3.3 Modify and Upload

Modify your Spark applications that you have run on your local machine so that they could be run on the Spark cluster. Then, upload your applications and input files to the cluster. You will need to copy the input files into HDFS or S3.

3.4 Run Spark Applications

Run your applications with the actual input files that are located in the HDFS. Then, copy-paste the result into `output.txt`.

3.5 Analyze Spark Applications

In this part, you will be using Spark's web UI, <master-dns>:8080, to analyze how the Spark cluster executes your applications (e.g. distribute workloads among worker nodes). Write a few paragraphs including snapshots from the web UI.