CSE427S

HOMEWORK M

M. Neumann

Due: THU 6 DEC 2018 4PM

Getting Started

Update your SVN repository.

When needed, you will find additional materials for *homework x* in the folder `hwx`. So, for the current assignment the folder is `hwM`.

Hint: You can **check your submission** to the SVN repository by viewing `https://svn.seas.wustl.edu/repositories/<yourwustlkey>/cse427s_fl18` in a web browser.

SUBMISSION INSTRUCTIONS

WRITTEN:

- all written work needs to be submitted electronically in *pdf format*[1] via GRADESCOPE
- provide the following information on *every* page of your `pdf` file:
  - name
  - student ID
  - wustlkey (your `wustlkey` indicates the location (SVN repository) for your code submissions; without this information we will not be able to grade your code!)
- start every problem on a *new page*
- **FOR GROUPS**: make a **group submission** on GRADESCOPE and provide the following information for **all group members** on every page of your `pdf` file: names, student IDs, and location of code submissions (**one** student's `wustlkey`)[2]

CODE:

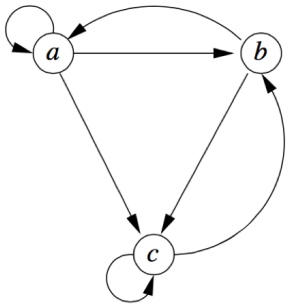
- code needs to be submitted via SVN repository commit (detailed submission instructions are provided whenever code submissions are required)
- make sure to always use the required *file name(s)* and *submission format(s)*
- **comment your code** to receive maximum credit

[1] Please, **type your solutions** or use **clear hand-writing**. If we cannot read your answer, we cannot give you credit nor will we be able to meet any regrade requests concerning your writing.

[2] It is sufficient to commit code to one SVN repository. If you do not specify the repository for your group's code submission clearly, we will only check the first repository, sorting wustlkeys alphabetically.

Problem 1: PageRank (20%)

In this problem you will do some more PageRank computations for the following graph:



You can use your SPARK code from **Lab 9** or do the computations by hand. In order to recieve full credit show your computations or add the (relevant) SPARK operations to your written answer.

(a) Compute the PageRank of each page (a, b, c) assuming no *teleportation*.[3]

(b) Compute the PageRank of each page (a, b, c) assuming $\alpha = 0.8$.[4]

Problem 2: PageRank on Real Webgraph (55%)

In this problem you will execute PageRank on subset of the a real webgraph (california.txt) provided in your `hwM` folder (in the SVN repository). It was constructed by expanding a 200-page response set to the search engine query "california". Note that, this data was collected some time back, so a number of the links will be broken by now. The graph has 9,664 nodes and 16,149 edges. Put the data into HDFS .
The data is formatted as follows:

- lines starting with ## are comments
- lines starting with n are nodes in the form of nodeID url (separated by space)
- lines starting with e are links in the form of nodeID nodeID (separated by space)

Find more implementation details on the Lab9 webpage: `https://sites.wustl.edu/neumann/courses/cse427s/labs/lab-9/`.

(a) Write a SPARK application called `PageRank.py/scala` based on the PageRank.pyspark/scalaspark code provided in Lab9 with the following additional implementation details:

- Modify your PageRank implementation to load and parse this data into RDDs. You want to create a node RDD to lookup the urls in the end and a links RDD for your PageRank computation.
- Handle print outs and commandline arguments as described in Part IV of Lab9.

[3] This is essentially Exercise 5.1.1 MMDS (page 175)
[4] This is essentially Exercise 5.1.2 MMDS (page 176).

- Change the stopping criteria of your algorithm to stop after 200 iterations. Add a checkpoint after every 50 iterations (instead of 5) to your PageRank program.
- The output of your job should be a ranking of webpages, i.e., the urls and ranks webpages sorted most important to least important.

(b) What does *checkpointing* do and why is is beneficial? Where are the checkpointed RDDs stored?

(c) Execute your SPARK application in local mode and analyze your application execution:

- Provide the command you used to launch the application.
- How many cores were used in the execution?
- How many stages and tasks did get executed?
- How many stages are in **one** PageRank iteration?

(d) Check the results. What are the 10 most important and the 10 least important webpages (urls) and their ranks.

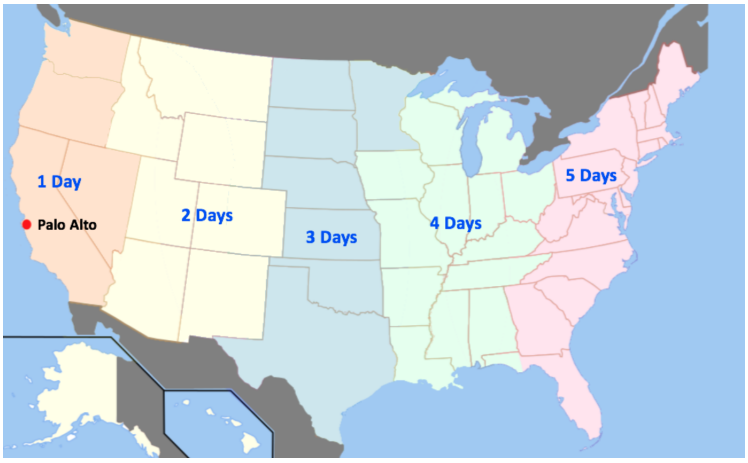
(e) Look them up in the web browser and verify the correctness of your algorithm. Do your result make sense intuitively?

Submit your implementation (answers to (b) by adding the `PageRank.py/scala` program to the `hwM` folder in your SVN repository. **To add the .py/scala file to your SVN repo and commit it run:**

```
$ svn add PageRank.*
$ svn commit -m 'hwM problem 2 submission' .
```

Problem 3: Best Location for Distribution Center (25%)

Dualcore's online advertising campaign is attracting many new customers, but many of them live far from *Dualcore's* only distribution center in Palo Alto, California. All shipments are transported by truck, so an order can take up to five days to deliver depending on the customer's location.



To solve this problem, *Dualcore* will open a new distribution center/warehouse to improve shipping times. The ZIP codes for the three proposed sites are 02118, 63139, and 78237. To decide on the best location you will compute the average distance between each of these locations and the locations of *Dualcore's* customers and select the location with the lowest average distance. The directory for this problem is:

```
~/training_materials/analyst/exercises/extending_pig
```

(a) As we cannot compute distances between ZIP codes, we have to transfer them into latitude/longitude pairs. Find a tab-delimited file mapping ZIP codes to latitude/longitude points here: `~/training_materials/analyst/data/latlon.tsv` . Put this file into HDFS. Observe and run the `create_cust_location_data.pig` script and explain what it does (consider input, output, and data processing steps in your explanation).

(b) Create a data set containing the ZIP code, latitude, and longitude for the possible warehouse locations. You can use this command:

```
egrep '^02118|^63139|^78237' ~/training_materials/analyst/data/latlon.tsv \
> warehouses.tsv
```

Put this file into HDFS and add the entries to your written answer.

(c) Compute the average distance between each warehouse location and the locations of the *Dualcore* customers based on the latitude/longitude information. Fortunately, you can use a PIG user defined function (UDF) called `HaversineDistInMiles` to achieve this.

This UDF is distributed with DataFu (`datafu-pig-incubating-1.3.1.jar`[5]). Download the `.jar` and add it to the following folder in your VM:

```
/usr/lib/pig/
```

Use the stubs provided in the `calc_average_distances.pig` script to implement your solution for the task. Here is an example on how to use the UDF in PIG :

```
REGISTER '/usr/lib/pig/datafu-*.jar';
DEFINE DIST datafu.pig.geo.HaversineDistInMiles;
places = LOAD 'data' AS (lat1:double, lon1:double, lat2:double, lon2:double);
dist = FOREACH places GENERATE DIST(lat1, lon1, lat2, lon2);
```

(d) Which of the three proposed ZIP codes has the lowest average mileage to *Dualcore's* customers?

Submit your answers to (c) by adding the `calc_average_distances.pig` script to the `hwM` folder in your SVN repository. **To add the .pig file to your SVN repo and commit it run:**

```
$ svn add calc_average_distances.pig
$ svn commit -m 'hwM problem 3 submission' .
```

[5] `https://repository.apache.org/content/groups/public/org/apache/datafu/datafu-pig-incubating/1.3.1/datafu-pig-incubating-1.3.1.jar`

Bonus Problem (5% up to a max. of 100%) - no group work!

Write a review for this homework and store it in the file `hwM_review.txt` provided in your SVN repository (and commit your changes). This file should only include the review, **no other information** such as name, wustlkey, etc. Remember that you are not graded for the content of your review, solely it's completion.

Provide the star rating for `hwM` via this link: `https://wustl.az1.qualtrics.com/jfe/form/SV_aaAc0ILmBleyKzj`.
You can only earn bonus points if you write a meaningful review of **at least 50 words** and provide the corresponding star rating. Bonus points are given to the **owner of the repository only** (no group work!).

Submit your review in the file **`hwM_review.txt`** provided in the `hwM` folder in your SVN repository. **To commit the file run:**

```
$ svn commit -m 'hwM review submission' .
```

Take 1 minute to provide a star rating for this homework – your star rating should match