Assignment 6
Graph Analysis using SocialLite
Due: March 6, 11:59 pm

In this programming assignment, you will use SocialLite to analyze social graph data. You are advised to work in groups of two on this assignment.

You will be analyzing the DBLP Computer Science bibliography, which is a database listing over 2 million articles on computer science. (Incidentally, it was so named because it was originally a DataBase and Logic Programming bibliography site).

1 Setup

SocialLite is a parallel/distributed query language for big data analysis. On the http://socialite-lang.github.io/ website, there is a "Quick Start" section to help you get started.

Download and setup the environment by running the following commands:

```
$ git clone https://github.com/socialite-lang/cs243.git hw6
$ cd hw6
$ ./setup.sh
```

The environment is structured as follows,

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>README.md</td>
<td>basic information on setting up the environment.</td>
</tr>
<tr>
<td>code/</td>
<td>examples</td>
</tr>
<tr>
<td>data/</td>
<td>DBLP data used in this assignment</td>
</tr>
<tr>
<td>logs/</td>
<td>log information</td>
</tr>
<tr>
<td>setup.sh</td>
<td>setup script</td>
</tr>
<tr>
<td>socialite/</td>
<td>source code and executable of SocialLite</td>
</tr>
</tbody>
</table>

We have provided two examples to help you understand SocialLite, PageRank (pr.py) and Shortest Path (sp.py), in the code/ directory. The Shortest Path query computes “single-source shortest paths”, that is, the shortest paths to all possible destinations given a single source. In this case, we have set the source to “Monica S. Lam”. At the end of the query, the example asked specifically the shortest path from Monica S. Lam to Donald E. Knuth and Edsger W. Dijkstra, two luminaries in computer science. To test if your environment is successfully set up, run the Shortest Path query.

```
$ socialite/bin/socialite code/sp.py
```

You should be able to see the following output.

reading authorId.txt
reading coauthor.txt
reading paperCount.txt
running shortest-paths
[u’Monica S. Lam’, u’Kunle Olukotun’, u’Leonidas J. Guibas’, u’Donald E. Knuth’]
[u’Monica S. Lam’, u’James C. Browne’, u’John R. Rice’, u’Edsger W. Dijkstra’]

Note: “u ‘Monica S. Lam’” indicates that it is the unicode encoding of the string ‘Monica S. Lam’.

Please don’t hesitate to go to office hours or post on piazza for help if you encounter any problem during setup.

2 Problem 1: Maximum Influence Paths

Suppose you wish to connect with an author of a paper through his co-authors, or co-authors of co-authors etc. Some papers may have many authors as a result finding the shortest path of co-authors to make the introduction may not be the best move. You want to find the co-authors that are most influential. For example, a professor Ann has written 100 papers in her career, 5 of which with grad student Bob. Bob, on the other hand, has only written those 5 papers and nothing else. It is clear that Ann has a lot of influence over Bob, but not the converse.

Thus, we define the influence of an edge, from A to B, in the co-authorship graph as the number of papers A and B co-authored divided by the total number of papers B authored. The influence of a path is defined as the multiplication of the influences of all its edges. The maximum influence path is the path that maximizes the influence of the source node to the target node. Modify the single-source shortest-paths algorithm in sp.py to implement maximum influence paths.

After you finished the algorithm, find the maximum influence path from “Monica S. Lam” to “Donald E. Knuth” and the path from “Monica S. Lam” to “Edsger W. Dijkstra”. Report those two paths with their maximum influence. Besides those two, find another example pair of authors where the maximum influence path is different from the shortest path between them. You could try out some examples of authors that you know, for example, other professors in the computer science department. Report the shortest path and the maximum influence path with its influence.

3 Problem 2: An open-ended problem

In this problem, you get to pick an analysis and implement it in SociaLite and apply it to the DBLP bibliography. We have collected a list of graph algorithms in algo_list.txt in the github repository. You are free to pick one from any on the list.
If you want to implement an analysis algorithm that is not listed in the file, please discuss it with the TAs.

1. Is it possible to express the analysis succinctly in SociaLite? If so, write the analysis, apply it to the DBLP algorithm, and report the running time.

2. If it is not feasible, explain.

### 4 Submission

Below is a list of files that you should submit for this assignment.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mip.py</td>
<td>The maximum influence paths algorithm</td>
</tr>
<tr>
<td>mip_result.txt</td>
<td>The result of the maximum influence paths queries</td>
</tr>
<tr>
<td>someGraphAlgorithm.py</td>
<td>Your solution to the graph algorithm you chose for Problem 2</td>
</tr>
<tr>
<td>report.txt</td>
<td>A brief description of the analysis you did and the running time of your query OR an explanation of why the algorithm you chose is infeasible to implement in SociaLite</td>
</tr>
</tbody>
</table>

Please submit your solution by following the process below.

1. `mkdir submit`
2. copy all of the files required for submission as listed in the table to the submit directory.
3. `scp submit/ to myth`.
4. `login to myth and cd to submit/`.
5. `run`
   
   ```
   $ /usr/class/cs243/submit
   in this directory. OR, if you’re working with a partner, type
   
   $ /usr/class/cs243/submit partner_SUID
   ```

Only one submission is required per pair. You can submit as many times as you want. We’ll grade using the latest copy.