Fall 2021	**CSC 466:** Knowledge Discovery from Data	Alex Dekhtyar

Lab 6: PageRank and Link Analysis

Due date: Friday, December 3, 10:00pm.

This is a firm deadline. No extensions will be given.

Overview

In this assignment you will implement PageRank ranking algorithm and run it on a number of datasets.

Assignment Preparation

This is an **individual** programming assignment.

Data

There is a number of datasets available for this assignment. They are broken into two categories: the small datasets and the SNAP[1] dataset(s).

Small Datasets

Your implementation shall run in adequate time on any of the datasets from the "small datasets" list provided below. The results of running your implementation of PageRank on these datasets must be put in your report.

1. **NCAA-FOOTBALL.** This dataset contains information about **every single game** played by Division I teams in the 2009 NCAA regular football season (before bowls and championships started). A total of 1537 games was played, their results are documented in the dataset.
2. **KARATE.** This dataset describes a small social network consisting of members of a university karate club.
3. **DOLPHINS.** A social network of a group of dolphins as observed by researchers over a period of time.
4. **LES-MISERABLES.** A graph of co-occurrence of difference characters in the chapters of Victor Hugo's novel *Le Miserables*.

General Data Format. To simplify parsing, all small datasets are available to you in a uniform data format. This format makes most sense for the football games, but has some redundant information for other datasets.

The data is presented in a CSV (comma-separated values) format. All datasets but **NCAA-FOOTBALL** have four columns in the CSV file. **NCAA-FOOTBALL** adds a fifth column, which is optional for processing (it is only useful if you go for extra credit, and even then you can ignore it).

The generic data format is:

```
<Node1>, <Node1-Value>, <Node2>, <Node2-Value>
```

Each row in the format above represents a single edge in the graph. Here, `<Node1>` and `<Node2>` are **unique ids** of two nodes forming an edge. `<Node1-Value>` and `<Node2-Value>` are two numeric labels that can be associated with an edge. In most of the datasets, one, or both of these values are set to 0. If non-zero values are present, they can be used to determine the direction of a link (if the dataset represents a directed graph).

Specific instructions regarding the data format for each of the datasets are included below.

[1] Stanford Large Network Dataset Collection, https://snap.stanford.edu/data/

NCAA-FOOTBALL. The data format includes a fifth, optional column and is interpreted as follows:

```
<Team1>, <Team1_Score>, <Team2>, <Team2_Score>, <overtime>
```

Here, the node labels are the names of the NCAA football teams, and the node values are the number of points each team scored in a game. The graph is directed. For each game played, an edge starts at the team that lost the game and ends at the team that won it. The winning team is listed first in the row (it is `<Team1>`), but this is not a given - to determine direction of an edge you need to get the score. The fifth, optional column indicates if the game involved an overtime.

There is only one data file, `NCAA_football.csv`. A sample of entries from the file is below:

```
"Ball State", 48, "Northeastern ", 14,
"Central Michigan", 31, "Eastern Illinois", 12,
"Southeast Missouri State", 35, "Southwest Baptist", 28, (OT)
```

Due to conversion issues, some scores are stored as integers, and some — as strings (e.g., `" 17"`). Your parser should strip the second and the fourth column value of everything except for digits.

KARATE. The dataset represents an **undirected graph** of social interactions. (If X interacted with Y, then Y interacted with X). There are two CSV files: `karate.csv` and `karateDir.csv`. The former represents each interaction as two rows in the CSV file in order to maintain symmetry. The latter represents each interaction using a single row. The node values are always 0. The node labels are integers from 1 to 34.

DOLPHINS. Same type of graph, data format and file availability (`dolphincs.csv` and `dolphinsDir.csv`) as for the **KARATE** dataset. The only difference is that node labels (names of individual dolphins) are strings enclosed in double quotes.

LES-MISERABLES. The dataset represents an **undirected, edge-labelled** graph of co-occurrences of charactes in the novel. Node labels are strings representing character names. The first node value is the edge label, representing *the total number of chapters in the book, where the two characters co-occur*. The second node value is always 0. Two CSV files are available, `lesmis.csv` and `lesmisDir.csv`. The former file represents each co-occurrence with a pair of entries, e.g.:

```
"Napoleon",1,"Myriel",0
"Myriel",1,"Napoleon",0
```

The latter file uses only one of the two rows.

GML format. Additionally, all datasets except for `NCAA-FOOTBALL` are available in GML (Graph Markup Language) format. GML format allows for some flexibility in representing graphs (a variety of means can be used to encode meta-data about the graph, and node and edge labels). GML representations may contain extra information about the graphs. You are welcome, but are not required, to use the GML files provided to you.

All data is found on the Lab data page:

http://users.csc.calpoly.edu/∼dekhtyar/466-Fall2021/labs/lab06.html

SNAP Datasets

SNAP, a.k.a. Stanford Lage Network Dataset Collection (yes, I know, the acronym does not match) is a well-known collection of large graphs and other datasets. SNAP contains dozens of datasets. For this lab assignment, we are providing you a few datasets to play with.

WIKI-VOTE. A graph of wikipedia internal voting results with 7115 nodes. Each edge represents one user account voting for another user account in some wikipedia related elections (for administrative positions, I believe).

P2P-GNUTELLA05. Gnutella is a peer-to-peer file sharing network. The dataset, consisting of 8846 nodes and over 31000 edges represents peer-to-peer connections on Gnutella on August 5 2002. An edge means that one sever has connected/requested information from the other server.

SLASHDOT-ZOO-NOV6-2008. The website Slashdot at some point implemented an upvote/downvote system (friend/foe system). The dataset, containing over 77,000 nodes representing user accounts, contains information about the friend/foe links on the site as of November 2008. Each edge is labelled with a 1 (friend) or -1 (foe) label.

AMAZON-MAY03. This graph contains information about product co-purchases made on Amazon.com during the month of May 2003. It contains over 410,000 nodes, representing individual products sold on the site. A link from product A to product B is in the dataset if people purchasing product A also often purchase product B.

LIVEJOURNAL1. LiveJorunal is/was a social network/blogging site. The site allows for directional following/friendship relationships. There are over 4 million nodes (user accounts) in the graph (close to 5 million, in fact), and an edge from one user to another means a directional "following" relationship.

Data Format. All SNAP datasets have a data format different from our small datasets. The SNAP datasets look as follows:

```
# First few lines are comments describing the dataset and its size
# Nodes: #Nodes Edges: #Edges
# FromNodeId    ToNodeId
0       1
0       2
0       3
0       4
0       5
...
```

Here, the first few lines start with `"#"` and represent comments. Your parser shall ignore them. Each line after the initial comments contain a description of one edge: the source node in the first column and the target node in the second column.

Additionally, SLASHDOT-ZOO-NOV6-2008 dataset contains a third column, "Sign" which, for each edge is either 1 or -1. This column may be ignored by you, unless you are considering a modified version of PageRank.

Lab Assignment

Write a program that takes as input a data file formatted in the way described above, runs the PageRank analysis on the graph extracted from the input file and outputs the individual items (football teams, states, etc...) ranked in descending order of their computed PageRank together with the PageRank score and the rank.

The program, `pageRank.java` (for example, if you are using Java) or `pageRank.py` (if you are using Python) shall take as input the file name. It may also (if you want) take as input a flag specifying whether the dataset you are reading in represents a directed or undirected graph (some of you may find it useful, but it is not absolutely necessary so it is left up to you. Please specify in the README file if you have the flag and if it is mandatory for your implementation).

It should parse the input, create a graph structure from it in main memory and run a version of PageRank ranking algorithm to rank the nodes in the graph.

You may implement the version of PageRank that was discussed in the class. You may also use extra information available to you (in case of the football season data: the score of the game and whether there was an overtime; in case of the *Les Miserables* data, the number of chapters/"thickness" of each connection) and for the Slashdot Zoo dataset – the sign of the friend/foe link to adapt your PageRank computation.

Your program should output an ordered list of **all** nodes in the graph. It should print the rank and the PageRank score of each of them. For example, your program can output:

```
1  Mississippi with pagerank: 0.030110446720353286
2  Florida with pagerank: 0.02406493620983336
3  Utah with pagerank: 0.01609201202237497
4  Oklahoma with pagerank: 0.01531666186988849
```

as the first four items for the `NCAA_Football.csv` input file.

(note, **the actual results may vary from the sample output above**)

If the output is large - feel free to directly dump it to a file.

Additionally, your program shall time itself. Specifically, we are interested in collecting information about the following times:

1. Read time. The time it takes to read in the data from the input file and build the initial graph data structure.
2. Processing time. The time it takes for the PageRank process to compute the PageRank of each node in the graph.
3. Number of iterations. The total number of iterations it takes for PageRank to converge (this is, unless you are running PageRank on a preset number of iterations).

You shall establish the PageRank for the following datasets:

- **Small Datasets**: all of them!
- **SNAP Datasets**: Pick three. One dataset must be either AMAZON-MAY03 or LIVEJOURNAL.

Making PageRank Efficient

PageRank is computed iteratively. Because the data needed to compute PageRank fits very nicely inside a matrix, each PageRank iteration can be made to run pretty fast in Python using `numPy` matrix algebra operations, and, when necessary broadcasting vector operations on matrices.

Report

The assignment will be graded primarily based on the contents of your report. Your report shall be a word-processed document submitted, preferably, in PDF format, which contains the following information:

1. **Front matter.** Title, course number, lab number, names of team members.
2. **Implementation Overview.** A short text (a few paragraphs) describing the details of implementation of your version of the PageRank algorithm. If you made any changes, or added any options to the algorithm, describe them as well.
3. **Results.** For each dataset, include a subsection which contains the following information:
   (a) Any specific information about the settings (if any) used to run PageRank on this dataset.
   (b) A concise copy of the output produced by your program on the dataset. *Concise* means that if your dataset is large (the output is more than two pages long), you can trim the output to show only the top — the most important, according to your algorithm — items from the dataset. The output should be typeset in a contrasting font from the rest of the report (e.g., use a `typewriter-style evenly spaced font` for the output). **DO NOT USE ANY OUTPUT FIGURES WITH BLACK BACKGROUND!!!!!!!**
   (c) Any observations, comparisons you can make about the work of PageRank on the dataset. Essentially, I am interested in your opinion on whether PageRank really did discover the proper ranking of the items in the dataset.

4. **Overall summary.** Provide a short overall summary of the observed results. How did PageRank work? Did it produce good rankings for most of the datasets? What types of datasets did it work better with? What types of dataset did not not work well with?

5. **Performance evaluation.** Evaluate the runtime performance of your PageRank implementation. You may want to do it separately for **SNAP** and for **small** datasets. Use the size of each dataset (number of nodes, number of edges) as independent variables, and use time to build the graph in main memory, time to compute PageRank, and the number of iterations to convergence as dependent variables (you can also study the time it takes to perform a single iteration). Report the results in a form of a graph (a scatter plot or a line plot may work well) displaying the dependency of the performance of your implementation on the size of the problem. You can combine the two independent variables (number of nodes and number of edges) into a single problem size value (e.g., the number of bytes it takes to represent the input).

   Provide some thoughts on the observed trends.

6. **Appendix. README.** Attach your `README` file with instructions on compiling and running your program as an appendix to your report.

Deliverables and submission instructions

This lab has only electronic deliverables. Submit the following:

- Report (as a separate from the rest of your submission archive file).
- Source code for your program in an archive.
- `README` file (in addition to inserting it into the report) and any additional files you need.

Submit all electronic deliverables including the report via handin.
Section 01:

```
$ handin dekhtyar lab06-466-01 <files>
```

Section 03:

```
$ handin dekhtyar lab06-466-03 <files>
```