# CMDA 3634 Fall 2017 Homework 01

Kevin Jiang

September 16, 2017

You must complete the following task by 5pm on Thursday 09/14/17.

Your write up for this homework should be presented in a LaTeX formatted PDF document. You may copy the LaTeX used to prepare this report as follows

1. Sign up for a http://sharelatex.com account.
2. Click on this link
3. Click on Menu/Copy Project.
4. Modify the HW01.tex document to respond to the following questions.
5. Remember: click the Recompile button to rebuild the document when you have made edits.
6. Remember: Change the author

*Each student* must individually upload the following files to the CMDA 3634 Canvas page at https://canvas.vt.edu

1. `firstnameLastnameHW01.tex` LaTeX file.
2. Any figure files to be included by `firstnameLastnameHW01.tex` file.
3. `firstnameLastnameHW01.pdf` PDF file.

You must complete this assignment on your own.

**150 points will be awarded for a successful completion.**
**Extra credit will be awarded as appropriate.**

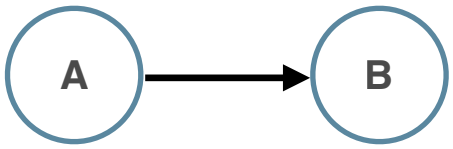
**Q1** *(20 points) Install a virtual machine on your computer and install Ubuntu.*

Twenty points will be awarded if you demonstrated a working Ubuntu VirtualBox to William Winter in Lecture 02.

---

**Q2** *(30 points) Understanding the PageRank Algorithm by example*

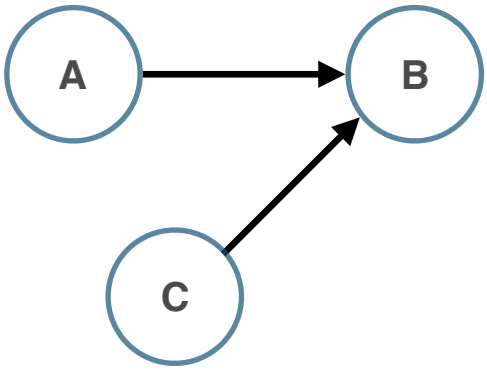
**Reliance**: Consider two web pages $A$ and $B$. If $A$ links to $B$, then we say $A$ relies on $B$ in some sense. We can express this action with a directed graph:




Since $A$ relies on $B$ but nothing relies on $A$, we say $B$ is more important than $A$.

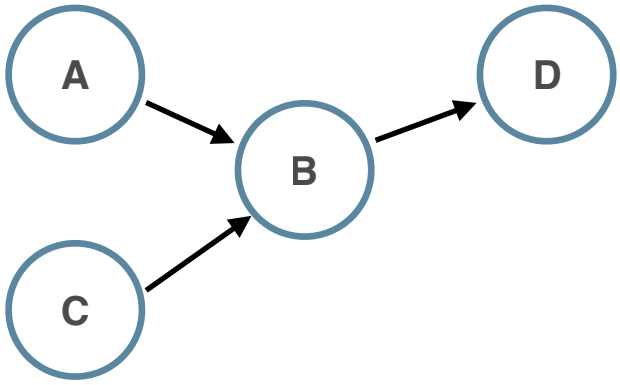
**Examples**: We further investigate this notion of reliance with some examples.

1: (reliance)



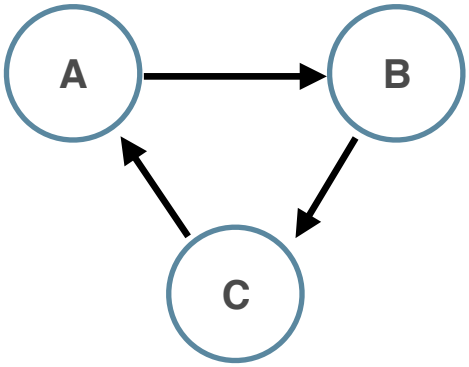

Both $A$ and $C$ rely on $B$, so we conclude $B$ is most important. No websites rely on $A$ or $C$, so we say they are equally important.

2 (transitivity):




Reliance is transitive. Because $A$ relies on $B$ and $B$ relies on $D$, $A$ also relies on $D$. Even though $D$ only has one page that directly relies on it, $A$ and $C$ also indirectly rely on it. So $D$ is most important in this example. $B$ is second most important as both $A$ and $C$ rely on it. Again, $A$ and $C$ are tied in importance (tied for least important in this case).

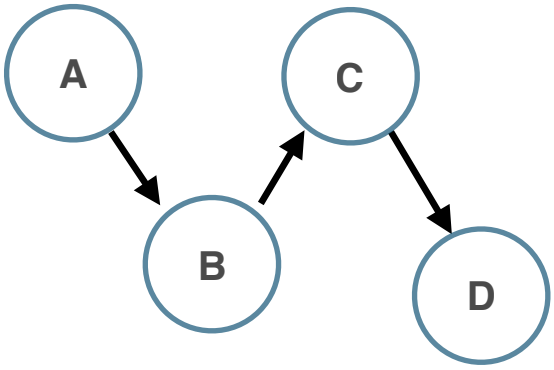
3 (circularity):




In reality, websites may form loops with their reliance. In this example, $A$, $B$ and $C$ are all equally important as they all are co-reliant.

**Exercises**:

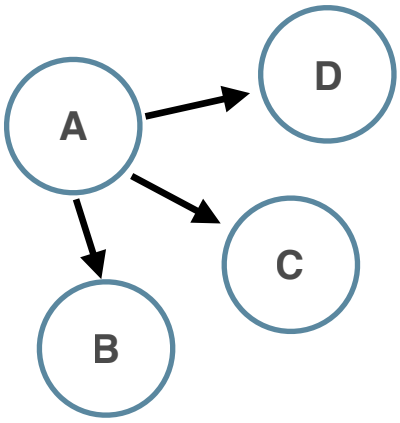
**Q2.1** (10 points)




For the above directed graph, give the importance hierarchy for the websites $A$, $B$, $C$ and $D$.

From greatest to least importance: $D$, $C$, $B$, $A$

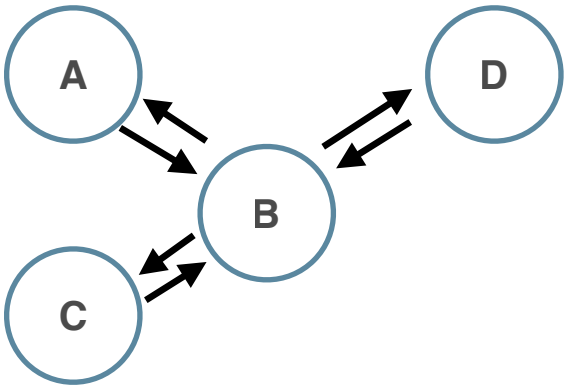
**Q2.2** (10 points)




We assume that a website that relies on multiple websites splits its reliance equally among those sites it relies on. For example, in the above graph $A$ relies on $B$, $C$ and $D$ equally. Then obviously $B$, $C$ and $D$ are equally important. What link could we add that would make $C$ the most important? (Remember: Links must have direction.) If no link would make $C$ most important explain why.

You could potentially add a link called $X$ where it is pointing directly to $C$ while both $B$ and $D$ would be pointing in the direction of $X$.

**Q2.3** (10 points)




Two websites may very well both link to each other. For the above directed graph, what is the importance hierarchy? Explain your reasoning.

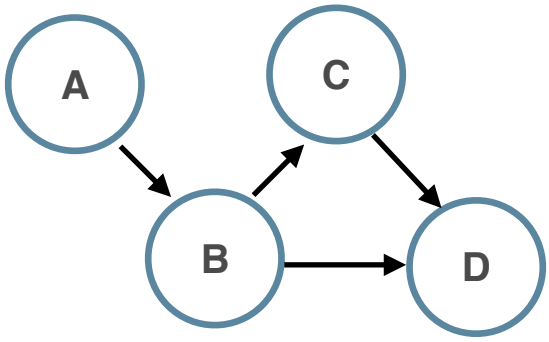
In this instance, link $B$ would be the most important hierarchy in this particular structure. This is because although B does point outwardly to $A$, $C$, and $D$, those links all lead directly to $B$.

---

**Q3** *(30 points) Translating a network into mathematical representation.*

We now need to turn a pictorial representation of a network into a mathematical object that can be understood by a computer.

**Example:** Consider the following network:




We now create an array to represent the network. Each row of the array corresponds to one node. The entries in the rows represent the nodes reliance relationships with other nodes. The array for the above network looks like this:

| | | | *To* | |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $A$ | 0 | 1 | 0 | 0 |
| $B$ | 0 | 0 | 1/2 | 1/2 |
| $C$ | 0 | 0 | 0 | 1 |
| $D$ | 0 | 0 | 0 | 0 |

Because $A$ relies on $B$ and only $B$, we put a one in the entry corresponding to $B$ in $A$'s row. (Remember: If a node relies on multiple other nodes the reliance is assumed to be split equally. Hence the $1/2$ in row $B$, column $C$.)

**Q3 Task:** For the networks in exercises 1, 2 and 3 of **Q2** give the array representation.

Exercise 1:

| | | | *To* | |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $A$ | 0 | 1 | 0 | 0 |
| $B$ | 0 | 0 | 1 | 0 |
| $C$ | 0 | 0 | 0 | 1 |
| $D$ | 0 | 0 | 0 | 0 |

Exercise 2:

| | | | *To* | |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $A$ | 0 | 1 | 1 | 1 |
| $B$ | 0 | 0 | 0 | 0 |
| $C$ | 0 | 0 | 0 | 0 |
| $D$ | 0 | 0 | 0 | 0 |

Exercise 3:

| | | | *To* | |
|---|---|---|---|---|
| | $A$ | $B$ | $C$ | $D$ |
| $A$ | 0 | 1 | 0 | 0 |
| $B$ | 1 | 0 | 1 | 1 |
| $C$ | 0 | 1 | 0 | 0 |
| $D$ | 0 | 1 | 0 | 0 |

---

**Q4** *(70 points) Starting to implement a network analysis tool.*

Modify the `network.c` skeleton code to add the desired functionality.

**Q4.1***(10 points) Designing a struct to contain the data from a network.*

First we need to design a struct to contain one of these networks. These networks can fully described through a few parameters and arrays. The struct should contain:

1. A variable representing the total number of nodes in the network.
2. A variable representing the maximum number of nodes any one node can be connected to.
3. A pointer to an array that contains the number of connections each node has.
4. A pointer to an array that contains the source of each connection in the network.
5. A pointer to an array that contains the destination of each connection in the network.
6. A pointer to an array of Page Rank values for each nodes. These Page Rank values will need to be of type doubles. (We will not use this on this homework, but you will need it soon.)

**Q4.2***(30 points) Reading data into a network struct.*

Now we will need a network reader helper method. The *networkReader* method should parse a comma separated data file and put the data into an instance of your newly defined Network struct.

The data file contains integers separated by commas. The first number is the total number of nodes. The second number is the maximum number of nodes a single node can be connected to. After these two values,

the remaining data represents connections. The remaining data is in pairs: the first number is the source of a connection while the second number is the destination of a connection. For example, lets look at what a data set may look like:

```
5, 4,
0, 1,
3, 1,
1, 0,
4, 3,
0, 2,
1, 3,
3, 4,
3, 0,
1, 4,
2, 3,
3, 2,
4, 1,
```

The 5 on the first row means there are five nodes in the network. The 4 on the first row means the maximum number of nodes any node can be connected to is four. (in a five node network, four is the highest the max connect value can be, though it will not always be one less than the number of nodes.) The second line indicates there is a connection from node 0 to node 1. The third line indicates there is a connection from node 3 to node 1. It continues similarly until the end of the data file.

Write a function `networkReader` that reads the data file, creates an instance of a network struct and initializes every member variable in the network struct (Except the vector of Page Rank, which should just be zeroed for now).

For the pointers to arrays in the network struct: create an array on the heap, fill in the array with data, and store the pointer to the array in the network struct.

**Q4.3***(30 points) Outputting information contained in the network.*

Lastly, we would like the main function to tell us something about the network. For a given data file and a desired node number $i$, the program should output how many connections node $i$ has and list the node numbers of all nodes $i$ is connected to.

The program is already set to take the name of the data file and a desired node (in that order). Now use the networkReader method from (b) to read the data. Then find the desired information and print the results neatly. Your input/output should look something like this:

```
> ./network numbers.csv 0
Node 0 is connected to 2 nodes.
Those nodes are:
1
2
```

Run your code on the supplied `numbers.csv` to find out how many connections node 1 has and which nodes 1 is connected to. For a data set this small, it would be easy to find the information by hand; keep in mind the program should be flexible enough to deal with various data files or different desired node numbers. Your code will be tested on a different data set with (possibly) a different value of $i$.

---

**Q5 (Extra Credit)***(Up to 20 points) Catching errors in a datafile.*

Sometimes there are errors in data files. Your program will be tested on some "defective" data sets. Write extra code that catches inconsistencies between the input data file and the file specification, deals with them, and throws appropriate errors messages/warnings/exceptions.

For example, one such inconsistency that *might* be tested for is feeding the program a data set with the same connection listed twice.

Add a brief description of what you did and why you did it to the report.

---