

# Lab 3: INFR 4710

Amirali Abari  
Faculty of Business and IT  
Ontario Tech University

## 1 Introduction

Welcome to the third lab! In this lab we will learn how to implement random walk for traversing graphs and how to parse data from a webpage. The lab instructions are set as minimum expectations. You are always encouraged to go beyond the expectations. Challenge yourself, have fun, and learn more!

Remember that when you have finished your lab activities, you need to show your final results to your TA. They will provide feedback if necessary and record your mark.

## 2 Setting Up

Download the modules `randomwalk.py` and `beautifulsoup.py` from Canvas. The Random Walk functions will be written in the `randomwalk.py` module. The webpage functions will be written in the `beautifulsoup.py` module. Then install BeautifulSoup<sup>1</sup>. For installation, you can use pip.

## 3 Tasks

In this lab, we focus on two different tasks. One is related on random walk on graphs, and another is focused on learning BeautifulSoup library for parsing web pages.

### 3.1 Task 1: Random Walk

Familiarize yourself with *random walks* for graphs. You might be interested in looking at this <http://www.lirmm.fr/~sau/JCALM/Josep.pdf> up to Markov Chains. With this information on hand, you are tasked with creating two Random Walk functions in the `randomwalk.py` module. One function will continue until you reach some destination node. The other function will run for a specific number of moves. The algorithm for Random Walk can be found below.

Set your maximum number of moves (or hops)

Set your destination node

Set your starting node as the current node

Add your starting node to the path

While you are not at the destination (or not at the maximum number of hops)

Randomly choose an adjacent node of the current node.

---

<sup>1</sup><https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Set the current node the chosen node.  
Add the the current node to the path

**Hint 1:** Check out the *random* package in Python for dealing with random choices.

**Hint 2:** Your new graph should be a MultiDiGraph instead of a normal Graph. The `add_node` and `add_edge` functions still work the same.

Now run your Random Walk Destination function 10 times and record the results to the following questions:

- How many movements did it take to reach the destination node?
- What was the most frequently visited node along the path? How often was it visited? (If there is a tie, list all of the ties)

Now experiment with your Random Walk Hops function and try to figure out the following questions (again, try to run your function 10 times):

- On average, how many hops does it take to discover the entire graph?
- How does this number compare to the average length of your Random Walk Destination function?

### 3.2 Task 2: BeautifulSoup

Your second task is to create a function that will take in a root webpage, find all of the hyperlinks on the webpage, and make a simple star graph<sup>2</sup> with the root webpage at the center.

To do this, you will be using BeautifulSoup and urllib3<sup>3</sup>. BeautifulSoup allows you to parse the HTML of a webpage and urllib2 allows you to connect to webpages. A simple refresher on HTML code can be found here: [https://www.w3schools.com/html/html\\_basic.asp](https://www.w3schools.com/html/html_basic.asp).

**Hint 3:** Web pages will contain 'http' in the url string.

Once you have created the function, change the root website to see the differences between different websites.

**Deliverable:** (1) Submit all your code to the Canvas. Your code should have internal documentation (or comments) explaining each line of code and instructions. (2) Write your answers to questions above in a document and submit to the Canvas.

---

<sup>2</sup>For more information on star graphs check here: [https://en.wikipedia.org/wiki/Star\\_%28graph\\_theory%](https://en.wikipedia.org/wiki/Star_%28graph_theory%28)

<sup>3</sup><https://pypi.org/project/urllib3/>