

T00: Graph Search

The first part of this assignment is very heavily based on Odest Chadwicke Jenkins and Jana Pavlasek's presentations given over Graph Search. [Presentation 1](#) by Odest Chadwicke Jenkins, [Presentation 2](#) by Jana Pavlasek. The assignment was assembled by

- This assignment should be done with your partner.
- You should seek help completing this assignment from the TAs at the evening lab.

Learning Objectives

- Practice breaking a larger problem down into smaller pieces using functions.
- Gain an understanding of graph search and one of the implementations of graph search.
- Develop understanding of the importance of data structures.
- Establish an understanding of the Breadth First Search algorithm and its purpose.

How to Start

- To begin, make a copy of this document and share it with all members of your team.
- Change the file name of this document to (for example, **neillz, martinj2 - Graph Search and Breadth First Search**).
- After you've copied the document, you can begin to work on it. The document saves periodically when connected to the internet.

Background

How are we able to tell robots to navigate from point A to point B? How do we tell them to avoid obstacles along the way? Graph Search is one solution to the problem of autonomous navigation. If we are able to plan a path from point A to point B, the robot should be able to traverse the path with ease.

The Application of Graph Search

Ultimately, we need to find an effective way to tell self-operating machines the best path to go. There are numerous other means of path traversal other than graph search. For example, we could move randomly. The most popular example of this is a roomba. Conversely, this algorithm would not help us with respect to efficiency. There is also the bug algorithm. [This presentation describes the bug algorithm](#). In essence, consider the following sourced from slide 2:

- Bug behaviors are simple:
 - Follow a wall (right or left)
 - Move in a straight line toward goal

While this is easy to code, there are some foundational issues. There are too many pitfalls, such as how we would need a known goal location as well as that the program itself is forgetful. That brings us to a third option: building a map. In essence, we can create a map and commit it to the autonomous machine's memory. This is where graph search comes in!

We can utilize graph search to traverse the map and compute the quickest path from a specified point and

Queue

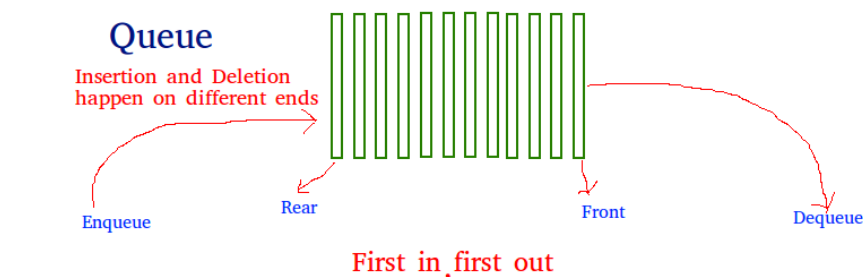
A queue is a data structure that adheres to the FIFO (First In, First Out) rule. This data structure is similar to a list in that it can store various types of data. However, it can best be thought of as a line in a supermarket. The first element (Person) added to the Queue will be the first element to “Pop”(Leave) off the queue (Checkout Line). We can use this data structure in some very helpful ways in regards to graph search, **storing nodes**.

How could the queue be useful when designing an algorithm?	
What data structure is similar to a queue? Why?	

Node

For Graph Search we will need to represent our data as a Node. Think of a node as a bubble of information that is connected to other nodes, which we can reference, while also altering the information stored in the node. This will be important later on when we get into how BFS works. Please also read the following article which briefly gives more information about what nodes are: <https://www.onlineschoolsreport.com/what-is-a-node-in-computer-science/>

First, let's take a look at the visual representation of the queue data structure.



[Queue image](#)

Let's say the strings: "Dog", "Cat", "Human", and "Horse" are all pushed into the queue in order, which one would be the first out of the queue?

Breadth First Search

Below is pseudocode for the Breadth First Search algorithm. This pseudo-code will help us understand how to perform breadth first search. With this rule set in mind we could design a program that can utilize Breadth First Search.

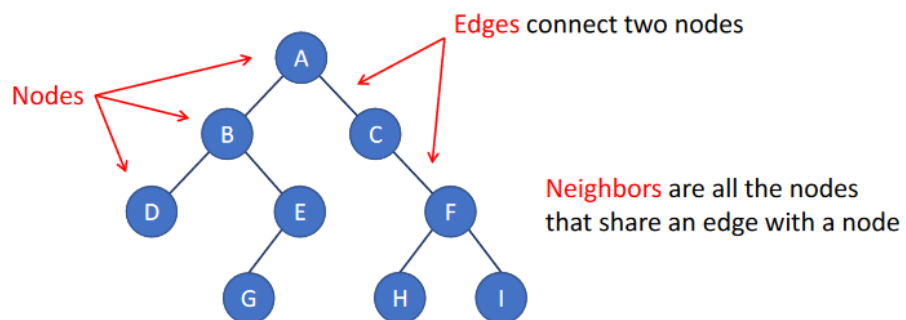
Breadth First Search Algorithm

1. Add start node to visit list. Set distance to zero.
2. Pop the first node from the front of visit list. Set as current node.
3. If current node is goal, go to 6.
4. Add each unvisited neighbor of current to back of visit list.
5. Set parent of each neighbor to current, set distance to current distance + 1. Go to 2.
6. Trace path backwards through parents.

BFS Pseudo-code

Analyzing the pseudo code above, why do we set distance to zero on step 1?	
What data structure can we use to make the visit list? Why?	

Below is an image of a tree, another data structure. A tree is a data structure composed of nodes, which we use to store information. The nodes here contain letters, which we will use for the following questions.



Node visualization

Using the pseudocode above and the image of the tree data structure, find a path from A to H.

Graph Search Implementation

Included in this assignment is an interactive Colab document that includes an example of graph search implementation. The reason for including the colab is to help visualize how graph search can be used.

The Link:

colab.research.google.com/drive/1ICtUIPZY9lySqQZ-cUulPmKmw5Mu7FI?usp=sharing

Follow the link **above**. Pay attention to the contents of the file. They should be printed out for you, but here is a screenshot in case something goes awry.

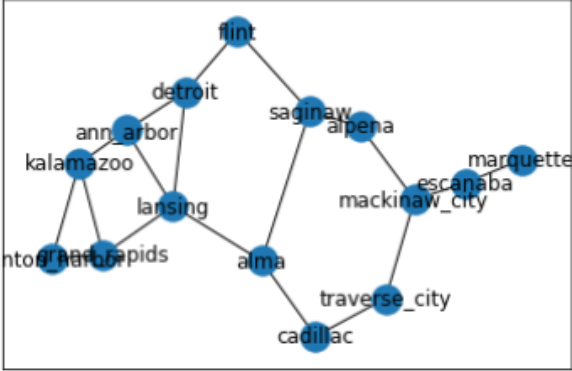
The file looks like:

```
benton_harbor kalamazoo 52
kalamazoo ann_arbor 99
ann_arbor detroit 42
benton_harbor grand_rapids 80
grand_rapids kalamazoo 50
grand_rapids lansing 67
lansing ann_arbor 65
lansing detroit 90
alma lansing 54
flint detroit 80
flint saginaw 40
alma saginaw 38
cadillac alma 84
traverse_city cadillac 48
saginaw alpena 148
traverse_city mackinaw_city 116
alpena mackinaw_city 94
mackinaw_city escanaba 147
escanaba marquette 66
-----
```

What do you notice about the contents of the file? What are the numbers associated with the city names? (Hint: Google Edge Costs)

Run the next code block; it should output something like the image below:

(note: node placement is inconsistent. If you see nodes and edges, it is working properly)

 <p>What is this? How did we populate this tree?</p>	
<p>Run the final code block. What does it return? Look back at the Breadth First Search pseudocode, and describe why it returned this result.</p>	
<p>You can edit the following code to find the quickest path from one city to another. Try replacing ann_arbor with detroit.</p> <pre>nx.dijkstra_path(g, "ann_arbor", "marquette", "")</pre> <p>What is the result?</p>	
<p>What is the difference between Dijkstra's Algorithm and Breadth First Search?</p>	

Teamwork Review

The objective of the following questions is to review the content provided within this teamwork and the necessary related concepts. If you have trouble responding to the questions, you may look back and reread through the sections.

<p>When analyzing the BFS pseudocode, what could you improve to reduce the number of searches it would require to find a path?</p>	
<p>In your own words, what is the definition of Graph Search?</p>	
<p>When thinking about Graph Search, what applications do you think it has to the world of autonomous robotics?</p>	

Submission Instructions

1. Submit the link to Moodle before the deadline.