Sean Poston

CS300 Homework 10

5/7/2020

Depth-First Search and Breadth-First Search

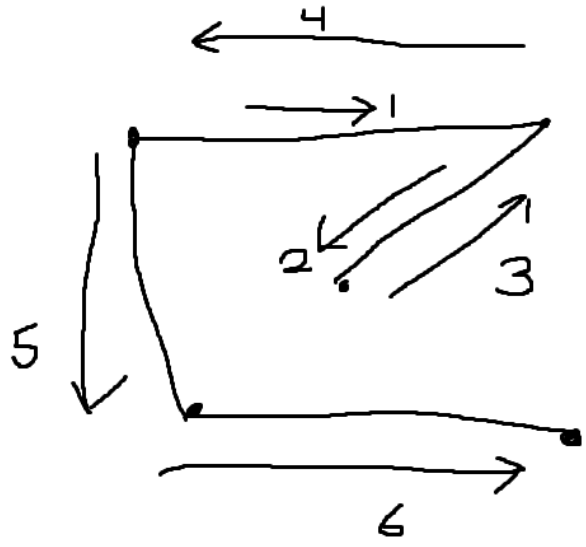
Chapter 28 focuses heavily on the intricacies of graphs and their inner workings. What this chapter tries to do is educate the reader on how to properly understand basic graph theory with its terms and procedures. The book covers the different terms one needs to know (vertices, edge, unweighted graph, weighted graph, etc.), the logic behind how these graphs work, and how to properly program them.

A large portion of this chapter is the details of how to program the graphs. It gives multiple ways of doing it, but mostly these ways consist of having multiple arrays with the name of the vertices, or points on the graph, in one graph and another graph with the edges, or the connections between the different vertices. These are then placed in a *Graph* object that stores the data and is able to fluently manipulate it.

One such manipulation of this data that the object can offer is a search. While there are many different ways to complete this, two of the most popular ways are depth-first search (DFS) and breadth-first search (BFS). These search methods are often used to search data trees as well.

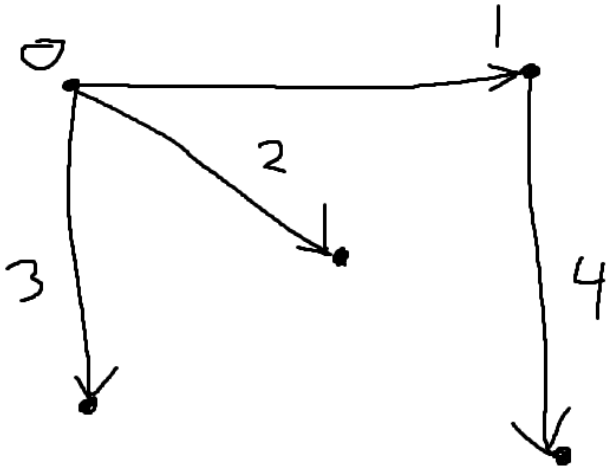
In a depth-first search, the algorithm starts at any given vertex and travels along a random path "deeper" into the graph. When the tracker comes to a dead end, it will recursively travel back on the path that it previously went until another untraveled path is available. This happens *ad infinitum* until the algorithm has found what it's looking for or the entirety of the vertices in the graph have been visited.

The book recommends to create a new array and to call it *isVisited*. The point of this array is to keep track of all the vertices that have been visited, as to save resources and time as well as to avoid an infinite loop. It will be a Boolean array that flips from false to true for each visited vertex.



The other method of searching is the breadth-first search. This search also functions similarly to its data tree cousin. Whereas the depth-first search will recursively traverse the graph until it's visited all points, a breadth-first search will visit adjacent nodes to the starting point. It will then visit vertices adjacent to those initial "children" vertices. This will happen with the same search conditions as above: until it's found what it seeks or until it runs out of vertices to search.

This differs from depth-first search in the way that it traverses the graph. Breadth-first will bounce back and forth between a pivot node and all nodes adjacent to that until all adjacent nodes have been visited. Then it will (usually) go to the first node that was searched previously and do the same bouncing back and forth for that node. This process is then continuously repeated.



While these advanced search methods aren't always useful for general searching, they are extremely handy for traversing linked nodes in data structures such as trees and graphs. I hope to learn further about these in the future.