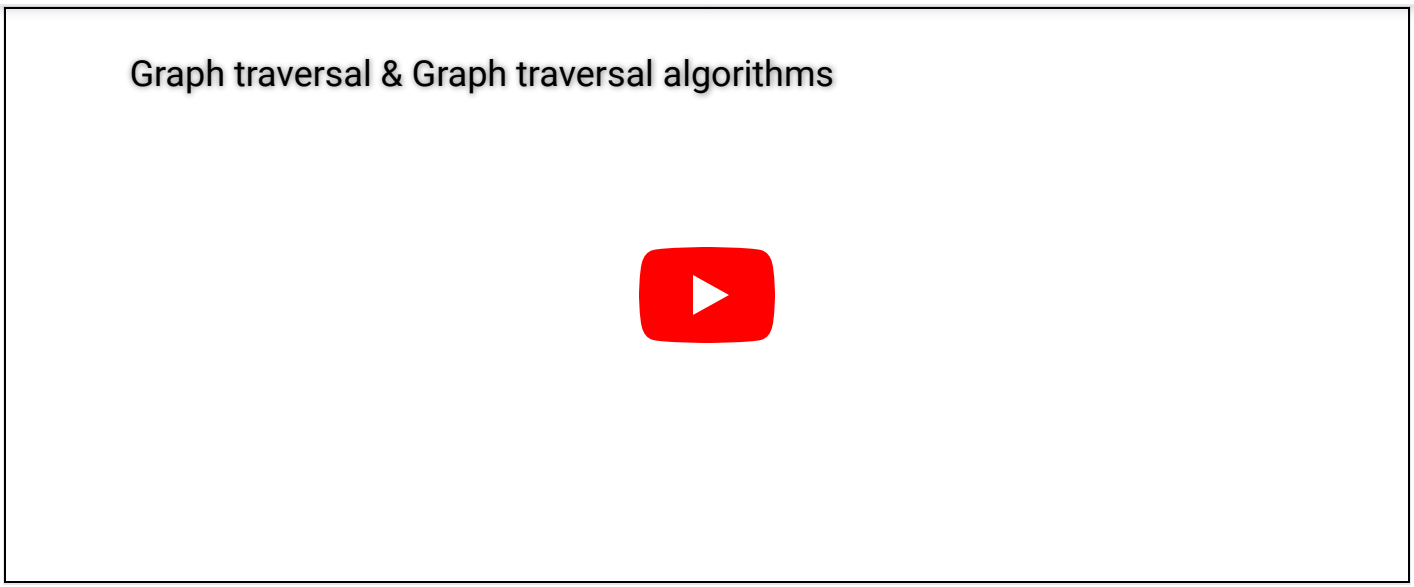


Show Course Contents

Overview Q&A Downloads Announcements

Graph traversal & Graph traversal algorithms

In the last lecture, we discussed in detail the different ways to represent a graph. We saw two of the most popular representation methods namely the adjacency list and the adjacency matrix. Today, we'll be interested in learning what graph traversal is and about different graph traversal algorithms. We'll also look at the reasons why these traversal techniques are so important.

What is graph traversal?

Graph traversal refers to the process of visiting (checking and/or updating) each vertex (node) in a graph. A smaller graph seems relatively easy to traverse. But when it comes to traversing a graph with a huge number of nodes, the process needs to be automated. Doing things manually increases the chances of missing some vertices or so. And visiting nodes of a graph becomes important when you need to change something for some nodes, or just need to retrieve the value present there or something else. And this is where our graph traversing algorithms come to the rescue.

Sequences of steps that are used to traverse a graph are known as graph traversal algorithms. There are two algorithms that are used for traversing a graph. They are:

1. Breadth-First Search (BFS)
2. Depth First Search (DFS)

We'll see these algorithms in much detail in the coming lectures, but for today, we'll just sketch out a rough idea about these algorithms.

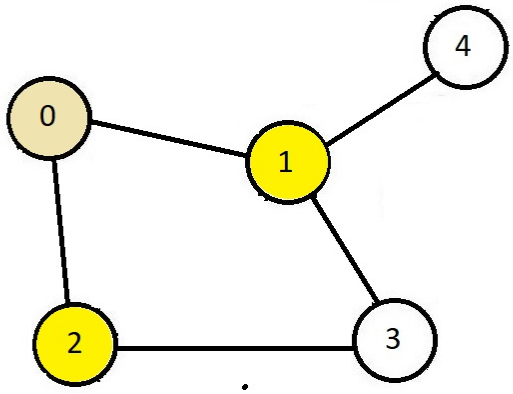
Breadth-First Search (BFS)

There are a lot of ways to traverse a graph, but the breadth-first search says, start with a node, and first, visit all the nodes connected to this node. But before we actually delve deep into this, there are a few terminologies we need to familiarise ourselves with.

Exploring a vertex (node):

- In a typical graph traversal algorithm, we traverse through (or visit) all the nodes of the graph and add it to the collection of all visited nodes.
- Exploring a vertex in a graph means visiting all the connected vertices.

Follow the illustration of the graph I have mentioned below. We'll use this for understanding the breadth-first search.



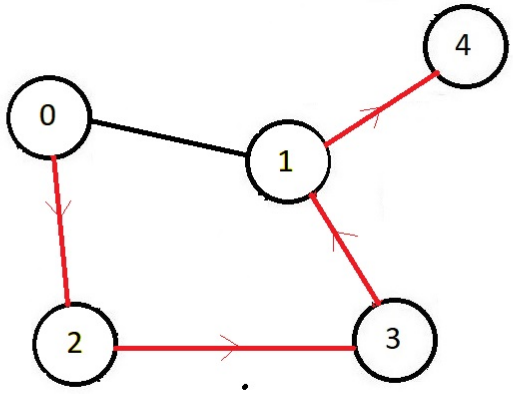
In breadth-first search, we use the queue data structure, and let me tell you that we won't be implementing the queue data structure all over again, rather we'll presume that we have all its utility functions pre-defined. Here, suppose we start with node 0 and put it in the list of visited nodes, then we visit nodes 2 and 1 since they are directly connected to node 0. And then gets visited node 3 and 4. So, the order of exploring would be, 0, 2, 1, 3, 4.

In breadth-first search, we visit from left to right all the nodes which lie on the same level. Here, node 0 was on one level, followed by nodes 2 and 1, and then

nodes 3 and 4.

Depth First Search (DFS)

Follow the illustration of a graph I have mentioned below. We'll use this for understanding the depth-first search.



In depth-first search, we use the stack data structure, and we'll not implement a stack either. Suppose they are pre-defined and available for us to use. Here, suppose we start with node 0 and put it in the list of visited nodes, then we visit node 2 and then visit nodes further connected to node 2. So, we visit node 3 and since it is further connected to node 1 and node 1 is connected to 4, we'll follow them in the same order. So, the order of exploring would be, 0, 2, 3, 1, 4.
In depth-first search, we visit from top to down all the nodes which are connected with each other. Here, node 0 was the parent, which is connected to node 2, which is again connected to node 3, and then node 1 and then node 4.
So, this was the basic idea of each of these two algorithms. We'll discuss them in great detail in the upcoming lectures. We'll first understand the algorithm, then followed by their applications and the places where we prefer one over the other.

Thank you for being with me throughout the session. I hope you enjoyed it. If you appreciate my work, please let your friends know about this channel too. Lectures on graphs have just started, and if you haven't saved this already, you just have to move on to codewithharry.com or my YouTube channel to access them. See you all there in the next lecture where we'll see the Breadth-First Search algorithm. Till then keep learning.

Previous

Next

CodeWithHarry