*AL\_Lab 05: Graphs and Uninformed Searches*

* Classes
* Graph Data Structure
* DFS, BFS of Graph



Making graph in python:

To represent graph data structures in Python, all we need to use is a dictionary where the vertices (or nodes) will be stored as keys and the adjacent vertices as values.

small\_graph **=** {

'A': ['B', 'C'],

'B': ['D', 'A'],

'C': ['A'],

'D': ['B']

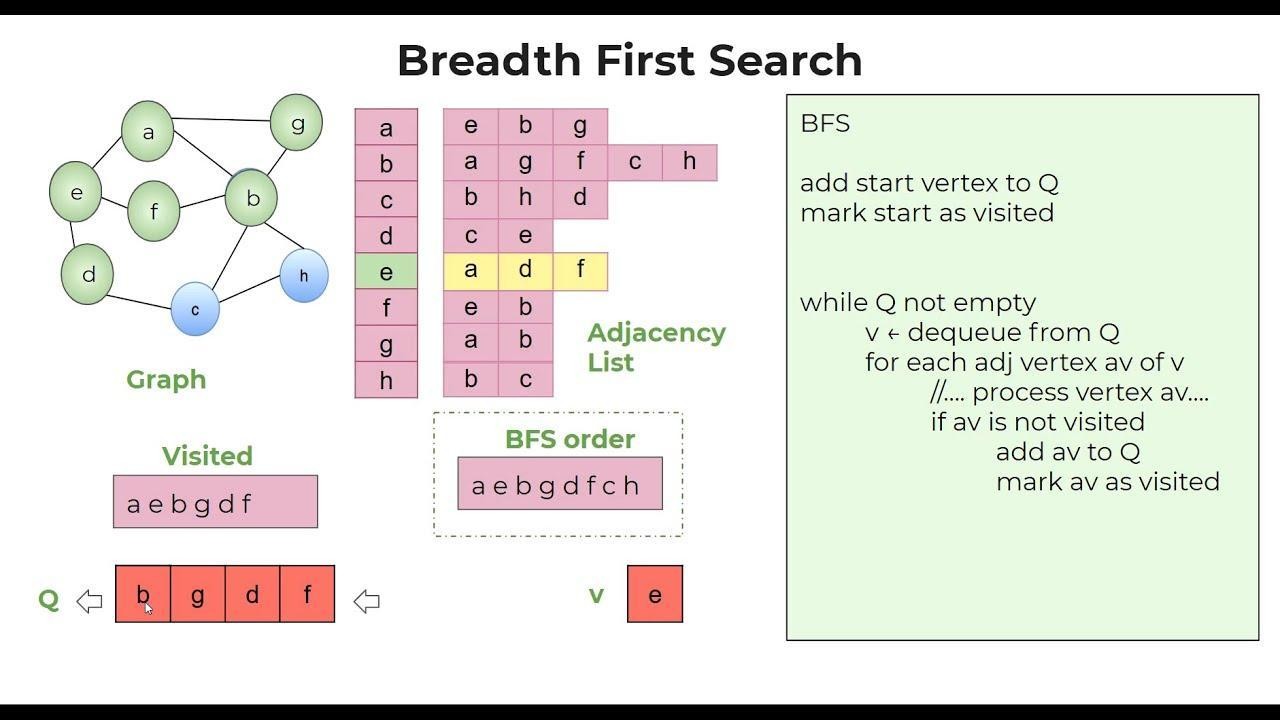
}



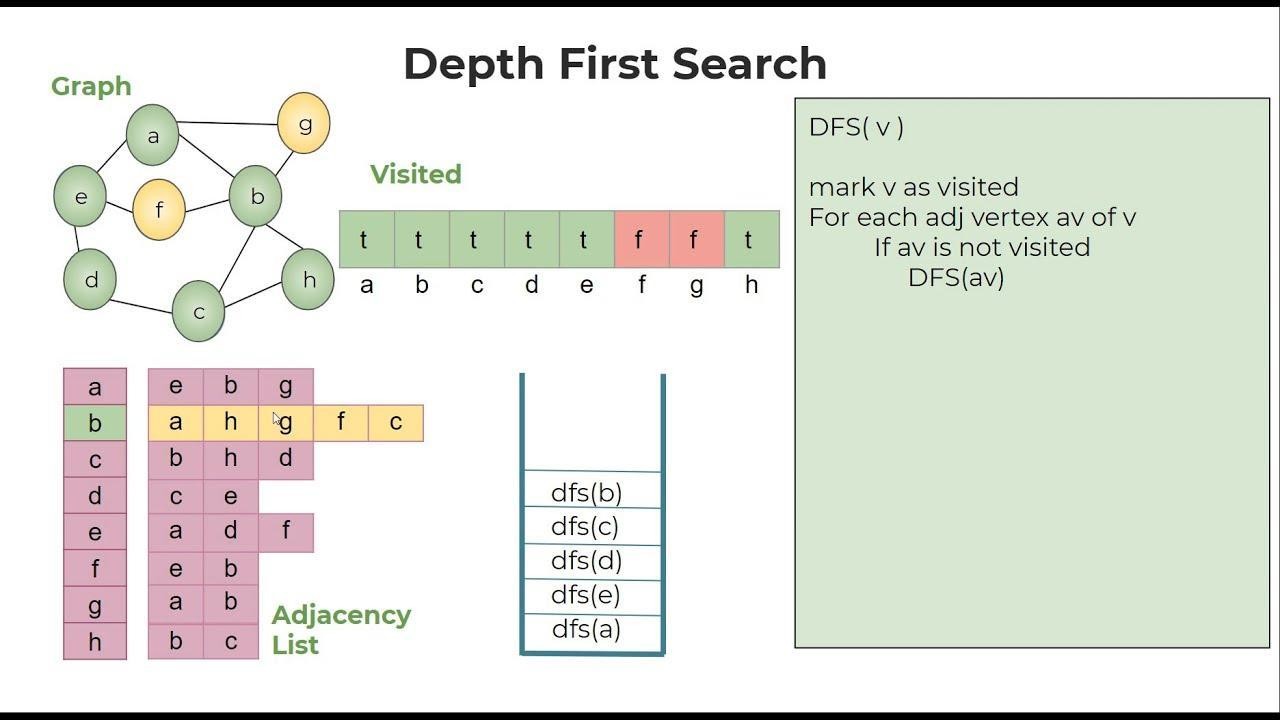
Algorithm of Graph traversal

1. Breadth First Search (BFS):

Breadth-First Search is a “blind” algorithm. It’s called “blind” because this algorithm doesn’t care about the cost between vertices on the graph. The algorithm starts from a root node (which is the initial state of the problem) and explores all nodes at the present level prior to moving on to the nodes at the next level. If the algorithm finds a solution, returns it and stops the search, otherwise extends the node and continues the search process. Breadth-First Search is “complete”, which means that the algorithm always returns a solution if exists. More specifically, the algorithm returns the solution that is closest to the root, so for problems that the transition from one node to its children nodes costs one, the BFS algorithm returns the best solution. In addition, in order to explore the nodes level by level, it uses a [queue data structure,](https://python.plainenglish.io/queue-data-strucure-theory-and-python-implementation-e58f3582c390) so new nodes are added at the end of the queue, and nodes are removed from the start of the queue. The pseudocode of the BFS algorithm is the following.



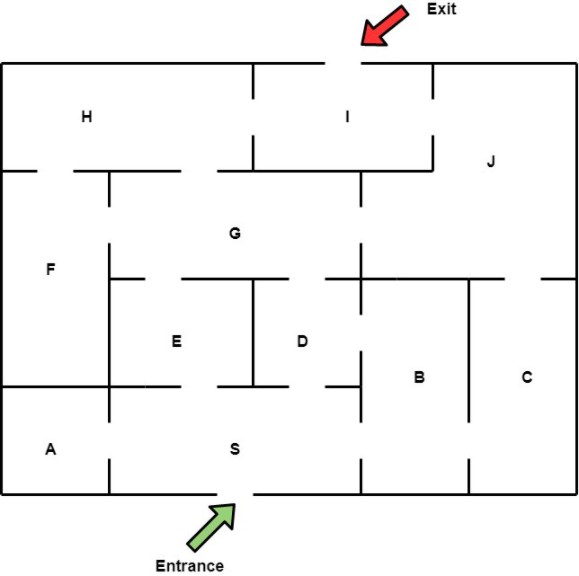
1. Depth First Search (DFS):



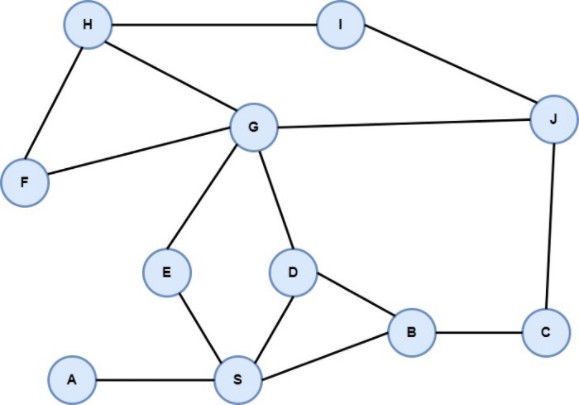
Lab Tasks

Task 01: MAZE FINDER USING BFS

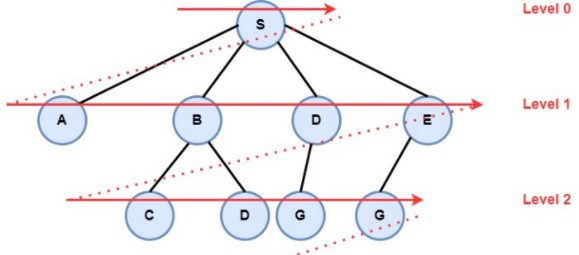
Suppose there is a maze such as an image shown below, and we want to navigate from the entrance to the exit with the less possible movements. As a movement, we consider each movement from one room to another. In our example, the maze consists of eleven rooms each of them has a unique name like “A”, “B”, etc. So, our goal is to navigate from room “S” to “I”.



After defining the problem, it’s time to model it into a graph. A very common way to do this is to create a *vertex for each room* and *an edge for each door* of the maze. After this modeling, the graph consists of 11 vertices and 15 edges as it seems below.



So, from each vertex we can navigate to its neighbors, starting from vertex “S” which is the initial state until the vertex “T” which is the target node of the problem. The BFS algorithm will explore all nodes at the present level prior to moving on to the nodes at the next level, as it seems in the following image.



# Task 02: Solve the same problem with DEPTH-FIRST SEARCH ALGORITHM

We will implement the DFS algorithm to find a path.