Lesson 2: Depth and Breadth First Search Algorithms

Notes

Book acknowledgment:

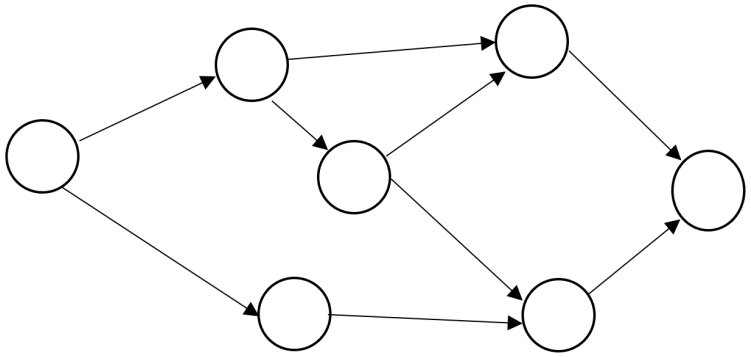
Goals

- Breadth First Search
- Depth First Search

1 Try it on your own

Given some graph $G = (V, E)$, how would you systematically explore and number all the connected nodes in V?

Try it on this example:



Write down your steps below:

2 Search Algorithms

Given s as the source node, the two search algorithms we study in this lession will systematically explore the edges of G to *discover* every node that is reachable from s. They also calculate the fewest number of edges from s to each reachable node in G. While very similar, these two search algorithms differ in implementation and analysis.

2.1 Breadth First Search

You can think of the Bread First Search (BFS) algorithm that explores each *layer* of the network

To simplify this algorithm, the BFS signifies each node as *nil*, *discovered*, and *explored*.

- All nodes start as nil and may later be explored.
- The BFS will keep up with a queue Q that keeps up with the list of nodes that should be explored next.

The BFS constructs a BF ***tree***, initially containing only its root node s. Whenever the search encounters a nil node v in the course of scanning the adjacency list of an already encountered node u, the node v and the edge (u, v) are added to the tree. Thus, we refer to the node u as the ***parent*** of v. Since a node is discovered at most once, then each node has at most one parent.

Written Steps:

Initialization: Create queue $Q = \emptyset$. Add source node s to Q. Mark s as explored.

1. If Q is empty, then proceed to Step 3. Otherwise, remove node v from the bottom of Q. Proceed to Step 2.
2. For some adjacent node $w \in V$ of v, if w is not visited, then add w to the bottom of Q, mark w as explored, and set the parent of w to be v. Otherwise, repeat Step 2.
3. Terminate the algorithm with the BF tree resulting from the parent vector of each node in V.

Pseudocode:

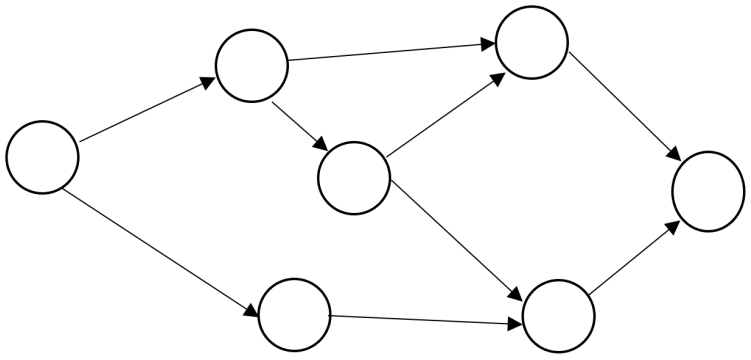
What do you think are the pros and cons of the BFS algorithm?

Questions?

Algorithm 1 Determine a BF tree for the Graph G with source node s

```
Let Q be the queue
Add s to Q.
Mark s as explored.
while Q is not empty do
  Remove v from Q
  for all adjacent nodes w ∈ V of v do
    if w is not explored then
      Add w to Q
      Mark w as explored
      Mark v as the parent of w
    end if
  end for
end while
```

Run the DFS algorithm on the following graph:



3 Depth First Search

Given the name of the Depth First Search (DFS), how do you think it differs from the BFS?

Just like the BFS algorithm, the DFS algorithm also constructs a ***tree***, initially containing only its root node s. Rather than exploring the *layers* of the graph (as with the BFS algorithm) by exploring **all** adjacent nodes of each node before moving to another node, the DFS instead continually explores a series of adjacent nodes until the algorithm reaches a node having no adjancent nodes.

The DFS algorithm is a repetitive algorithm that *backtracks* to a recently visited node.

Written Steps:

Initialization: Create stack $S = \emptyset$. Add source node s to the top of S. Mark s as explored.

1. If S is empty, then proceed to Step 3. Otherwise, remove node v from the top of S. Proceed to Step 2.

2. For some adjacent node $w \in V$ of v, if w is not visited, then add w to the top of S, mark w as explored, and set the parent of w to be v. Otherwise, repeat Step 2.

3. Terminate the algorithm with the DF tree resulting from the parent vector of each node in V.

Pseudocode:

Algorithm 2 Determine a DF tree for the Graph G with source node s

```
Let S be a stack.
Insert s at the top of S.
Mark s as explored.
while S is not empty do
  Remove v from the top of S
  for all adjacent nodes w ∈ V of v  do
    if w is not explored then
      Add w to the top Q
      Mark w as explored
      Mark v as the parent of w
    end if
  end for
end while
```

What do you think are the pros and cons of the BFS algorithm?

Questions?

Run the DFS algorithm on the following graph: