CSE 180 **Algorithmic Thinking**

About Finding Paths in Graphs

12 May 2021 *Teach One Another*

Introduction

Finding paths in graphs is one of the main applications of graph theory. Various graph-searching algorithms come into play, the best known of these are depth-first search and breadth-first search.

Depth-First Search

```
ALGORITHM DFS(G)
// Implements a depth-first search traversal of a given graph
// Input: Graph G = <N, L>
// Output: Graph G with its nodes marked with consecutive integers
// in the order they are first encountered by the DFS traversal
mark each node in N with 0 as a mark of being "unvisited"
count ← 0
for each node n in N do
        if n is marked with 0
                dfs(n)

dfs(n)
// visits recursively all the unvisited nodes connected to node n
// by a path and numbers them in the order they are encountered
count ← count + 1
mark n with count
for each node m in N adjacent to n do
        if m is marked with 0
                dfs(m)
```

Breadth-First Search

```
ALGORITHM BFS(G)
// Implements a breadth-first search traversal of a given graph
// Input: Graph G = <N,L>
// Output: Graph G with its nodes marked with consecutive integers
// in the order they are visited by the BFS traversal.
mark each node in N with 0 as a mark of being "unvisited"
count ← 0
for each node n in N do
        if n is marked with 0
                bfs(n)
bfs(n)
// visits all the unvisited nodes connected to node n
// by a path and numbers them in the order they are visited
count ← count + 1;
mark n with count and initialize a list with n
while the list is not empty do
        for each node m in N adjacent to the first node in the list do
                if m is marked with 0
                        count ← count + 1
                        mark m with count
                        append m to the list
        remove the first node from the list
```

For Practice

Here is a picture of a small Eleven-Node Seventeen-Link Graph that you can practice tracing through these algorithms with.

The Next Level

If the links have weights (lengths, distances) attached to them, then finding a minimum-length path is a little more work than if they don't. But even unweighted links can be thought of as having a default weight of one, so that a minimum is achieved by just finding a path with the lowest number of links.

Your Tasks

TODO Set up for Practice

- Label each node of the graph.
- Create a list of each node's adjacent (neighboring) nodes.

TODO Trace DFS Algorithm

- List the nodes in the order they are visited.

TODO Trace BFS Algorithm

- List the nodes in the order they are visited.

TODO Set up for the Next Level

- Label each link with a random weight.
- Decide if DFS or BFS (or some other algorithm) is most appropriate to use.
- Trace a shortest path from a node on one side of the graph to a node on the other side.