Graph Search Algorithms

Today we'll examine some graph traversal algorithms at a high level, focusing on two very common ones: Depth First Search and Breadth First Search.

Graph Representations

Recall that a graph is a collection of nodes/vertices connected by edges.

As an adjacency matrix:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

As an adjacency list:

1: 2, 5
2: 1, 5
3: 2, 4
4: 3, 5, 6
5: 1, 2, 4
6: 4

DFS

Take the first path you see. Once you've gone as far as you can, backtrack and try the most recent alternatives. Programmatically, we can traverse a graph using a stack. Here is some pseudocode for how to carry out a DFS traversal in graph G, starting at vertex v (assuming there are no cycles so no revisiting)

DFS(G, v):
    stack S = stack_new()
    push(S, v)
    while !stack_empty(S)
        u = pop(S)
        for w in neighbors(u, G)
            push(S, w)
BFS

Take all the paths - don’t go way ahead down one path while leaving others in the dust. Programmatically, we traverse a graph using a queue. Here is some pseudocode for how to carry out a BFS traversal in graph $G$, starting at vertex $v$ (assuming there are no cycles so no revisiting).

DFS($G$, $v$):
    queue $Q = $ queue_new()
    enq($Q$, $v$)
    while !queue_empty($Q$)
        $u = $ deq($Q$)
        for $w$ in neighbors($u$, $G$)
            enq($Q$, $w$)
    END

Reflection

Notice how similar BFS and DFS are. They operate in almost the exact same way, differing only in the order of traversal. In the mazes above, try to get through using each algorithm. What sort of algorithm do you use normally?