Considering the same graph lets look at breadth-first traversal (BFS).

Diagram

Description automatically generatedText

Description automatically generated with low confidence

In BFS we would visit the nodes horizontally, i.e. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. For example, we initially start at node 1, and work our way across the [1] row in the matrix. That means we visit 2, 3, and 4. The next row is 2, so we work across the [2] row. Then [3] and so on. When we hit the end of the matrix we have processed every node horizontally.

Alg.: BFS(startNode,adjacencyList)  
    traverse(startNode) // do whatever you need to at the node;  
    for col in 1..12  
        if matrix[startNode][adjacentNode] == 1 then  
            adjacencyList.add(adjacentNode)  
    if !(adjacencyList.isEmpty()) then  
        adjacentNode <-- adjacencyList.removeNext()  
        BFS(adjacentNode,adjacencyList)  
// done

A hand trace looks like this

Table

Description automatically generated with low confidence

The recursive BFS returns all the way back to the first call when the adjacencyList becomes null (i.e. empty). If the graph is not a directed graph (you can travel in either direction on an edge) then we must include an attribute at the node that indicates whether we have already been there in a previous visit. This is also true in a cyclic graph since we don't want to cross an edge more than once in the BFS (or DFS for that matter).