2) Algorithm dfs(root)
Input: The root of the tree
output: words printed in alphabetical order
words := an empty stack
push root to the stack
mark root as visited
while stack is not empty:
    node = words.pop()
    if node has no children:
        print ~~stack~~ words
    for child nodes of node:
        if child node not visited:
            mark child node visited
            push child node to words

4) Algorithm bfs(root)
Input: root node
Output: words printed sorted by length
words := an empty queue
enqueue root to words
mark root as visited
while queue is not empty:
    node = words.dequeue()
    if node has no children:
        print words
    for child node of node:
        if child node not visited:
            mark child node visited
            enqueue child node to words

5) Algorithm dfs (~~root~~ u, v, G)

Input : edge (u, v) and graph G

Output : True if edge is a bridge, false otherwise.

If u or v have no adjacent vertices:

    return false

path := empty stack

push u to path

mark u and v as visited

while stack != empty :

    ~~node~~ node = path.pop()

    if node is not u and adjacent vertex is v:

        return false

    for adjacent vertex of node :

        if adjacent vertex not visited :

            mark adjacent vertex visited

            push adjacent vertex to path

return true

$O(m+n)$