**Charles Isah Saidu: eHealth Job**

**Depth First Search Algorithm**

There are basically two types of Graph traversal algorithm. Breadth First Search (BFS) and Depth First Search (DFS). While BFS searches in levels by exploring (hence, visiting one) and all adjacent nodes from source vertex s before moving to the next vertex chosen from one of the adjacent vertex of s, Depth First Search is a Graph traversal algorithm that searches “deeper” in the graph whenever possible, synonymous to movement in a **Maze.**

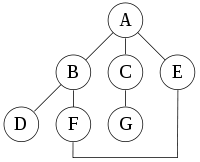


Figure 1: Graph: source wikipedia.org

Assuming in Figure One, a depth-first search starting at node A will visit the nodes in the following order A, B, D, F, E, C, G. That is, once a node or a vertex is visited, subsequent traversals will follow from that node/vertex until a dead-end is reach after which the algorithm backtracks to the previous node and follows that path.

As explain previously, a typical analogy of DFS is a walk in a maze. The maze consists of narrow passages (think of edges) and intersections where passages meet (vertices). Suppose that someone is lost in the maze. She knows there’s an exit and plans to traverse the maze systematically to find it. Fortunately, she has a ball of string and a marker pen. She starts at some intersection and goes down a randomly chosen passage, unreeling the string. At the next intersection, she goes down another randomly chosen passage, and so on, until finally she reaches a dead end. At the dead end she retraces her path, reeling in the string, until she reaches the previous intersection. Here she marks the path she’s been down so she won’t take it again, and tries another path. When she’s marked all the paths leading from that intersection, she returns to the previous intersection and repeats the process.

This behaviors can be conveniently be implemented using **Recursive Backtracking Algorithm.** However, most algorithm DSF algorithms tend to use iterative approach because of stack-overhead problems inherent in a typical recursive algorithm. A Typical DFS runs at O(E+V) provided the graph is represented by an adjacency list.

Some popular applications of DSF Algorithm

1. Topology sort problem
2. Web crawlers
3. Finding Strongly connected components

**Algorithm**

**Recursive Algorithm**

1 **procedure** DFS(*G*,*v*):

2 label *v* as discovered

3 **for all** edges from *v* to *w* **in** *G*.adjacentEdges(*v*) **do**

4 **if** vertex *w* is not labeled as discovered **then**

5 recursively call DFS(*G*,*w*)

**Non – Recursive algorithm**

1 **procedure** DFS-iterative(*G*,*v*):

2 let *S* be a stack

3 *S*.push(*v*)

4 **while** *S* is not empty

5 *v* = *S*.pop()

6 **if** *v* is not labeled as discovered:

7 label *v* as discovered

8 **for all** edges from *v* to *w* **in** *G*.adjacentEdges(*v*) **do**

9 *S*.push(*w*)