# Depth-First Search (DFS) Algorithm

## Introduction

Depth-First Search (DFS) is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It uses a stack data structure (explicitly or via recursion) to track the vertices to visit next.

## Graph Representation

We represent the graph using an adjacency list (dictionary in Python). Each key is a node, and the corresponding value is a list of neighboring nodes.

Example:  
```  
graph = {  
 'A': ['B', 'C'],  
 'B': ['D', 'E'],  
 'C': [],  
 'D': [],  
 'E': ['F'],  
 'F': []  
}  
```

## Algorithm Steps (Iterative)

1. Initialize an empty set `visited` and a stack containing the `start` node.  
2. While the stack is not empty:  
 - Pop the top node from the stack.  
 - If the node has not been visited:  
 - Mark it as visited and process it (e.g., print it).  
 - Push its neighbors onto the stack in reverse order (so the leftmost neighbor is processed first).  
3. Continue until all reachable nodes are visited.

## Pseudocode

```  
def depth\_first\_search(graph, start):  
 visited = set()  
 stack = [start]  
  
 while stack:  
 node = stack.pop()  
 if node not in visited:  
 print(node)  
 visited.add(node)  
 # Add neighbors in reverse to process in correct order  
 for neighbor in reversed(graph[node]):  
 stack.append(neighbor)  
```

## Example Walkthrough

Given the example graph above and starting at node 'A', the DFS order is:  
- Visit A  
- Push C, then B  
- Pop B → Visit B  
- Push E, then D  
- Pop D → Visit D (no neighbors to add)  
- Pop E → Visit E  
- Push F  
- Pop F → Visit F  
- Pop C → Visit C  
Final traversal: A → B → D → E → F → C

## Backtracking

Backtracking occurs when we reach a node with no unvisited neighbors. The algorithm then pops the next node from the stack, effectively backtracking to explore other branches.

## Conclusion

DFS is useful for:  
- Pathfinding when you need to explore one path fully before trying another.  
- Topological sorting, cycle detection, and solving puzzles with a single solution path.