

Topological Sort (DFS)

● MEDIUM

■ Introduction

Topological Sort is the ordering of tasks where dependencies come first. Example: to compile code, first compile dependencies. In a DAG (Directed Acyclic Graph), topological order ensures every node comes before its successors.

■ Core Intuition — DFS Post-order

Key Insight

- DFS and push nodes to a STACK after all their descendants are visited.
- A node is pushed to stack only after ALL its dependencies are processed.
- Finally, pop the stack to get topological order (reverse post-order).
- Only valid for DAGs — cycles make topological sort impossible.

■ Visual

Topological DFS Trace

DAG: $5 \rightarrow 2$, $5 \rightarrow 0$, $4 \rightarrow 0$, $4 \rightarrow 1$, $2 \rightarrow 3$, $3 \rightarrow 1$

DFS from 5:

visit 5 → visit 2 → visit 3 → visit 1 (no unvisited neighbors)

push 1 → stack:[1]

back to 3: push 3 → stack:[1,3]

back to 2: push 2 → stack:[1,3,2]

back to 5: visit 0 → push 0 → stack:[1,3,2,0]

push 5 → stack:[1,3,2,0,5]

DFS from 4: visit 4 → push 4 → stack:[1,3,2,0,5,4]

Pop stack: 4, 5, 0, 2, 3, 1 ← topological order

■ Java Implementation

Java

```
class Solution {
    public int[] topoSort(int V, ArrayList<ArrayList<Integer>> adj) {
        boolean[] visited = new boolean[V];
        Deque<Integer> stack = new ArrayDeque<>();
```

```

for (int i=0; i<V; i++)
if (!visited[i]) dfs(i, adj, visited, stack);

// Pop stack to get topological order
int[] result = new int[V];
for (int i=0; i<V; i++) result[i] = stack.pop();
return result;
}

private void dfs(int node, ArrayList<ArrayList<Integer>> adj,
boolean[] visited, Deque<Integer> stack) {
visited[node] = true;
for (int neighbor : adj.get(node))
if (!visited[neighbor]) dfs(neighbor, adj, visited, stack);
stack.push(node); // Push AFTER all descendants are processed
}
}

```

■ Time & Space Complexity

Case	Time	Space	Why
Time	$O(V + E)$	$O(V)$	DFS visits each node and edge once
Space	$O(V)$	$O(V)$	Visited array + stack both $O(V)$

■ Common Mistakes

Watch Out

- Pushing node to stack BEFORE recursion — must push AFTER (post-order).
- Running on cyclic graph — topological sort undefined for graphs with cycles.
- Not iterating all nodes — disconnected nodes get missed.

■ Practice Problems

Case	Time	Space	Why
Medium	LC 207	Course Schedule	Check if topo sort possible
Medium	LC 210	Course Schedule II	Return topo sort order

