Topological Sort Cheatsheet

Overview

- Topological sort is an algorithm for sorting a directed acyclic graph (DAG).
- It orders the nodes in a graph such that for every directed edge from node A to node B, node A comes before node B in the ordering.
- It is used in various applications, such as task scheduling and dependency resolution.

Algorithm

```
def topological_sort(graph):
    # Initialize variables
    visited = set()
    stack = []

    # Visit each node in the graph
    for node in graph:
        if node not in visited:
            topological_sort_helper(graph, node, visited, stack)

    # Return the topologically sorted nodes
    return stack[::-1]

def topological_sort_helper(graph, node, visited, stack):
    # Mark the current node as visited
    visited.add(node)

    # Visit each adjacent node
    for neighbor in graph[node]:
        if neighbor not in visited:
            topological_sort_helper(graph, neighbor, visited, stack)

    # Add the current node to the stack
    stack.append(node)
```

Time Complexity

- Worst-case performance: O(|V| + |E|)
- Best-case performance: O(|V| + |E|)
- Average-case performance: O(|V| + |E|)

Resources

- Topological Sort Wikipedia
- GeeksforGeeks: Topological Sort
- Visualgo: Topological Sort