**Topological Ordering Topsort**

**Iheb Gafsi**\*

INSAT Student

**Iheb.engineer@gmail.com**

**Definition:**

Topological sorting (or Topsort) is a linear ordering of the vertices of a directed graph such that for every directed edge (u, v), vertex u comes before vertex v in the ordering. In other words, it arranges the vertices in such a way that all directed edges go from left to right.

Topological sorting is only possible for directed acyclic graphs (DAGs). If the graph contains cycles (i.e., there is a path from a vertex back to itself), then no valid topological sorting exists.

**Use cases:**

Topological sorting has various applications, including task scheduling, dependency resolution, and finding the order of execution in projects or workflows. It is a fundamental algorithm used in graph theory and has practical use in various fields such as project management and software engineering.

**Algorithm:**

1. # Variables

2. graph = adjacency list

15.

16. # The Depth-First Search Algorithm

17. def dfs(arg, visited, graph):

18. if arg in visited:

19. return []

20. else:

21. visited.add(arg)

22. l = []

23. for i in graph[arg]:

24. l = dfs(i, visited, graph) + l

25. return [arg] + l

26.

27. # The Actual Topological Ordering Algorithm

28. def topsort(graph):

29. n = len(graph)

30. visited = set()

31. l = []

32. for i in range(n):

33. if not i in visited:

34. l = dfs(i, visited, graph) + l

35. return l

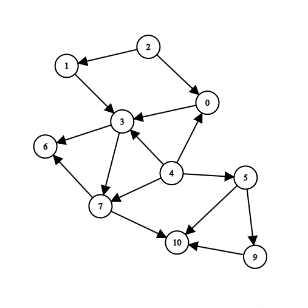
36.

37. print(topsort(graph))

38.

**Example:**

Here’s a small example illustrating an example of input outputs for the BFS:



We will use the Python code down below to outline the output of the algorithm on this graph:

1. # Variables

2. graph = [

3. [3],

4. [3],

5. [0, 1],

6. [6, 7],

7. [0, 3, 5, 7],

8. [9, 10],

9. [],

10. [6, 10],

11. [],

12. [10],

13. []

14. ]

15.

16. # The Depth-First Search Algorithm

17. def dfs(arg, visited, graph):

18. if arg in visited:

19. return []

20. else:

21. visited.add(arg)

22. l = []

23. for i in graph[arg]:

24. l = dfs(i, visited, graph) + l

25. return [arg] + l

26.

27. # The Actual Topological Ordering Algorithm

28. def topsort(graph):

29. n = len(graph)

30. visited = set()

31. l = []

32. for i in range(n):

33. if not i in visited:

34. l = dfs(i, visited, graph) + l

35. return l

36.

37. print(topsort(graph))

38.

The corresponding output is:

1. Python >> [8, 4, 5, 9, 2, 1, 0, 3, 7, 10, 6]

