UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN

Advanced Algorithms

Lab Experiment 2

MTECH-COMPUTER SCIENCE ENGINEERING
CYBER SECURITY AND FORENSICS

Name: Jigesh Sheoran
SAP ID: 590025428

Aim: To implement Topological Sorting for a Directed Acyclic Graph (DAG) and validate the correctness of the output.

Theory :

Topological sorting is a linear ordering of vertices of a DAG such that for every directed edge **u** → **v**, vertex **u** appears before **v** in the ordering. It is applicable only to DAGs; if a cycle exists, topological sorting is not possible.

Algorithm (Kahn's Algorithm – BFS based) :

1. Compute in-degree of each vertex.
2. Insert all vertices with in-degree 0 into a queue.
3. Remove a vertex from the queue, add it to the result list.
   Decrease in-degree of its adjacent vertices; if any becomes 0, insert into queue.
4. Repeat until the queue is empty.

Validation of Correctness :

- All vertices appear exactly once in the output.
- For every edge u → v node u appears before v in the order.
- If the algorithm cannot process all vertices, it correctly detects a cycle.

Time Complexity

O(V + E), where V is the number of vertices and E is the number of edges.

Code:

topological_sorting.py > ...

```python
# Author: Jigesh Sheoran
# Last Modified: 01/01/2026

from collections import defaultdict, deque

def topological_sort(vertices, edges):

    # create adjacency list and indegree dictionary
    graph = defaultdict(list)
    indegree = {v: 0 for v in vertices}

    for u, v in edges:
        graph[u].append(v)
        indegree[v] += 1

    # insert all vertices with indegree 0 to queue
    queue = deque()
    for v in vertices:
        if indegree[v] == 0:
            queue.append(v)

    topo_order = []

    # process vertices
    while queue:
        node = queue.popleft()
        topo_order.append(node)

        # reduce indegree of adjacent vertices
        for neighbour in graph[node]:
            indegree[neighbour] -= 1
            if indegree[neighbour] == 0:
                queue.append(neighbour)

    # check for cycle
    if len(topo_order) != len(vertices):
        print("\nGraph contains a cycle. Topological sorting is not possible.")
        return None

    return topo_order


# user input
print("Topological Sorting using Kahn's Algorithm\n")

n = int(input("Enter number of vertices: "))
vertices = []

print("Enter vertex names:")
for i in range(n):
```

```
    vertices.append(input(f"Vertex {i+1}: "))

e = int(input("\nEnter number of directed edges: "))
edges = []

print("Enter edges (u v) meaning u → v:")
for i in range(e):
    u, v = input(f"Edge {i+1}: ").split()
    edges.append((u, v))

result = topological_sort(vertices, edges)

if result:
    print("\nTopological Order:")
    print(" → ".join(result))

print("\nExperiment completed successfully.")
```

Output 1: Where graph is cyclic

```
sheoraninfosec@Jigeshs-MacBook-Air Exp 2 % /usr/local/bin/python3 "/Users/sheoraninfosec/Docume
h's MacBook Air/Masters UPES/Semester 2/Advanced Algorithms/Lab /Exp 2/topological_sorting.py"

Enter number of vertices: 6
Enter vertex names:
Vertex 1: A
Vertex 2: B
Vertex 3: C
Vertex 4: D
Vertex 5: E
Vertex 6: F

Enter number of directed edges: 7
Enter edges (u v) meaning u → v:
Edge 1: A B
Edge 2: A D
Edge 3: B E
Edge 4: C E
Edge 5: D F
Edge 6: F A
Edge 7: E F

Graph contains a cycle. Topological sorting is not possible.

Experiment completed successfully.
sheoraninfosec@Jigeshs-MacBook-Air Exp 2 %
```

Output 2: Where the graph is acyclic

```
sheoraninfosec@Jigeshs-MacBook-Air Exp 2 % /usr/local/bin/python3 "/Users/sheora
h's MacBook Air/Masters UPES/Semester 2/Advanced Algorithms/Lab /Exp 2/topologic
Topological Sorting using Kahn's Algorithm

Enter number of vertices: 5
Enter vertex names:
Vertex 1: A
Vertex 2: B
Vertex 3: C
Vertex 4: D
Vertex 5: E

Enter number of directed edges: 4
Enter edges (u v) meaning u → v:
Edge 1: A C
Edge 2: A B
Edge 3: C D
Edge 4: E A

Topological Order:
E → A → C → B → D

Experiment completed successfully.
sheoraninfosec@Jigeshs-MacBook-Air Exp 2 %
```

—----------END OF DOCUMENT —----------