practical no 2

Implement the DAG

he DAG (Directed Acyclic Graph) objective is commonly used in various domains like machine learning, optimization, and graph theory. Typically, a DAG is a graph where the edges have a direction, and there are no cycles (i.e., no paths where you can start and return to the same node).

To implement a "DAG objective," the specific context and the goal of the problem you're working on should be clarified. Here are a few possibilities:

1. **Topological Sorting**: A common problem with DAGs is to find the **topological order** of nodes. This is an ordering where for every directed edge u -> v, u appears before v in the ordering.
2. **Shortest Path in DAG**: In a weighted DAG, you might want to find the shortest path from a source node to other nodes.
3. **DAG-based Optimization Problems**: These may involve using the structure of a DAG to efficiently solve problems, such as job scheduling, resource allocation, or dynamic programming problems like finding the longest path or optimal ordering

import networkx as nx

import matplotlib.pyplot as plt

# Create a Directed Graph

dag = nx.DiGraph()

# Add nodes (tasks or steps in a workflow)

dag.add\_nodes\_from(["A", "B", "C", "D", "E"])

# Add directed edges (dependencies)

dag.add\_edges\_from([

("A", "B"), # A must be done before B

("A", "C"),

("B", "D"),

("C", "D"),

("D", "E") # D must be done before E

])

# Check if the graph is a DAG

is\_dag = nx.is\_directed\_acyclic\_graph(dag)

print(f"Is the graph a DAG? {is\_dag}")

# Topological Sort (to get a valid order of execution)

if is\_dag:

topo\_order = list(nx.topological\_sort(dag))

print(f"Topological Order: {topo\_order}")

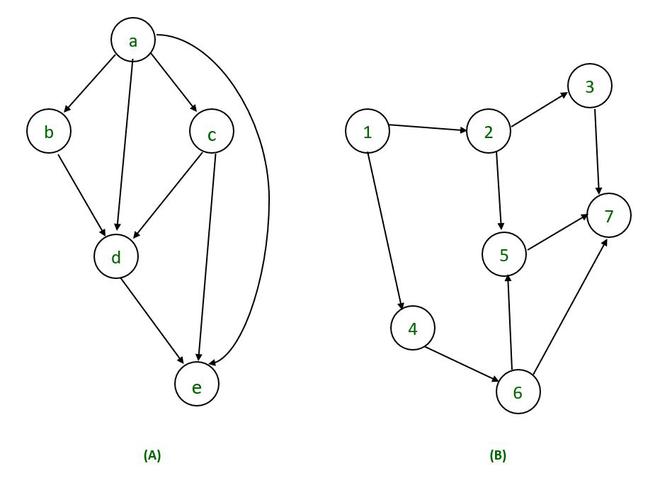
# Visualize the DAG

pos = nx.spring\_layout(dag)

nx.draw(dag, pos, with\_labels=True, node\_color='lightblue', edge\_color='gray', node\_size=3000, font\_size=10)

plt.title("Directed Acyclic Graph (DAG)")

plt.show()

**output** 

**learning outcomes**

 gain a clear understanding of the key properties of a Directed Acyclic Graph: directed edges, acyclic nature, and how they differ from general graphs.

 **Components of a DAG**: Understanding nodes, directed edges, and how they form paths without any cycles.

 **Visualization**: Ability to visualize DAGs and interpret various graph representations (e.g., adjacency matrix, adjacency list).