import networkx as nx

import matplotlib.pyplot as plt

# Activity list: (Activity Name, Start Node, End Node, Duration)

activities = [

    ('A', 1, 2, 11),  # Build Legs

    ('B', 1, 3, 3),   # Build Top

    ('C', 1, 4, 10),  # Build Drawer

    ('D', 4, 5, 3),   # Paint Drawer

    ('E', 3, 5, 1),   # Paint Top

    ('F', 2, 6, 2),   # Paint Legs

    ('G', 5, 6, 1),   # Attach Drawer to Top

    ('H', 6, 7, 1)    # Attach Legs

]

# Create a directed graph

G = nx.DiGraph()

# Add edges with durations and activity names

for name, start, end, duration in activities:

    G.add\_edge(start, end, activity=name, duration=duration)

# Forward Pass - Calculate earliest start and finish times

earliest\_start = {}

earliest\_finish = {}

for node in nx.topological\_sort(G):

    incoming = G.in\_edges(node, data=True)

    if not incoming:

        earliest\_start[node] = 0

    else:

        earliest\_start[node] = max(earliest\_finish[u] for u, \_, \_ in incoming)

    earliest\_finish[node] = earliest\_start[node] + max((data['duration'] for \_, \_, data in incoming), default=0)

# Backward Pass - Calculate latest start and finish times

latest\_finish = {}

latest\_start = {}

# Start from the end node

end\_node = max(earliest\_finish, key=earliest\_finish.get)

latest\_finish[end\_node] = earliest\_finish[end\_node]

latest\_start[end\_node] = latest\_finish[end\_node]

for node in reversed(list(nx.topological\_sort(G))):

    outgoing = G.out\_edges(node, data=True)

    if not outgoing:

        latest\_finish[node] = earliest\_finish[end\_node]

    else:

        latest\_finish[node] = min(latest\_start[v] for \_, v, \_ in outgoing)

    latest\_start[node] = latest\_finish[node] - max((data['duration'] for \_, \_, data in outgoing), default=0)

# Identify critical path

critical\_path = []

for u, v, data in G.edges(data=True):

    est = earliest\_start[u]

    lst = latest\_start[u]

    if est == lst:

        critical\_path.append(data['activity'])

# Display results

print("Critical Path Activities:", " → ".join(critical\_path))

print("Project Completion Time:", earliest\_finish[end\_node], "hours")

# Draw the network graph

pos = nx.spring\_layout(G)

edge\_labels = {(u, v): f"{d['activity']} ({d['duration']}h)" for u, v, d in G.edges(data=True)}

nx.draw(G, pos, with\_labels=True, node\_color='lightblue', node\_size=1500, font\_size=10, font\_weight='bold')

nx.draw\_networkx\_edge\_labels(G, pos, edge\_labels=edge\_labels)

plt.title("Project Network Diagram - Critical Path Method")

plt.show()