**Name: Om Chandrakant Mahajan Roll No:88**

**Practical Name: Branch and Bound Search Batch: B3**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

from queue import PriorityQueue

class Node:

def \_\_init\_\_(self, state, cost, path):

self.state = state

self.cost = cost

self.path = path

def \_\_lt\_\_(self, other):

return self.cost < other.cost

def branch\_and\_bound(graph, start, goal):

visited = set()

priority\_queue = PriorityQueue()

priority\_queue.put(Node(start, 0, [start]))

while not priority\_queue.empty():

current\_node = priority\_queue.get()

current\_state = current\_node.state

if current\_state == goal:

return current\_node.cost, current\_node.path

if current\_state in visited:

continue

visited.add(current\_state)

for neighbor, cost in graph[current\_state]:

if neighbor not in visited:

priority\_queue.put(Node(neighbor, current\_node.cost + cost, current\_node.path + [neighbor]))

return None, None

graph = {

'S': [('A', 3), ('B', 6)],

'A': [('D', 2), ('C', 7), ('S', 3)],

'B': [('E', 6), ('D', 3), ('S', 6)],

'C': [('A', 7), ('G', 0)],

'D': [('E', 2), ('B', 3), ('A', 2), ('G', 4)],

'E': [('D', 2), ('B', 6), ('G', 1)],

'G': [('D', 4), ('E', 1), ('C', 0)]

}

start\_node = 'B'

goal\_node = 'E'

result\_cost, result\_path = branch\_and\_bound(graph, start\_node, goal\_node)

if result\_cost is not None and result\_path is not None:

print("Shortest path:", result\_path)

print(f"Shortest path cost from {start\_node} to {goal\_node}: {result\_cost}")

else:

print("No path found from start node to goal node.")

**OUTPUT:**

**C:\Users\comp\om\venv\Scripts\python.exe "C:/Users/comp/om/branch and bound.py"**

**Shortest path: ['B', 'D', 'E']**

**Shortest path cost from B to E: 5**

**Process finished with exit code 0**