TASK.2

import heapq

# Define the graph as an adjacency list

graph = {

'A': {'B': 15, 'C': 20},

'B': {'A': 15, 'C': 10, 'D': 30},

'C': {'A': 20, 'B': 10, 'D': 25, 'E': 5},

'D': {'B': 30, 'C': 25, 'E': 15},

'E': {'C': 5, 'D': 15}

}

def dijkstra(graph, start, end):

pq = [(0, start, [])] # (cost, node, path)

visited = set()

while pq:

(cost, node, path) = heapq.heappop(pq)

if node in visited:

continue

path = path + [node]

visited.add(node)

if node == end:

return (cost, path)

for neighbor, weight in graph[node].items():

if neighbor not in visited:

heapq.heappush(pq, (cost + weight, neighbor, path))

return (float("inf"), [])

# Example usage

distance, path = dijkstra(graph, 'A', 'E')

print(f"Shortest distance from A to E: {distance}")

print(f"Path: {' -> '.join(path)}")

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.