CN Assignment

Akshat Patil

AIDS-A

Roll: 10

prn : 12311353

Task: Write a program to implement:

a. Network Routing: Shortest path routing

import heapq

class NetworkRouter:

def \_\_init\_\_(self):

self.graph = {}

def add\_connection(self, from\_node, to\_node, weight):

if from\_node not in self.graph:

self.graph[from\_node] = []

if to\_node not in self.graph:

self.graph[to\_node] = []

self.graph[from\_node].append((to\_node, weight))

self.graph[to\_node].append((from\_node, weight)) # Undirected network

def dijkstra(self, start):

distances = {node: float('inf') for node in self.graph}

distances[start] = 0

priority\_queue = [(0, start)]

previous\_nodes = {node: None for node in self.graph}

while priority\_queue:

current\_distance, current\_node = heapq.heappop(priority\_queue)

if current\_distance > distances[current\_node]:

continue

for neighbor, weight in self.graph[current\_node]:

distance = current\_distance + weight

if distance < distances[neighbor]:

distances[neighbor] = distance

previous\_nodes[neighbor] = current\_node

heapq.heappush(priority\_queue, (distance, neighbor))

return distances, previous\_nodes

def shortest\_path(self, start, end):

distances, previous\_nodes = self.dijkstra(start)

path = []

current = end

while current is not None:

path.insert(0, current)

current = previous\_nodes[current]

return path, distances[end]

# Example Usage

router = NetworkRouter()

router.add\_connection('A', 'B', 2)

router.add\_connection('A', 'C', 5)

router.add\_connection('B', 'C', 1)

router.add\_connection('B', 'D', 3)

router.add\_connection('C', 'D', 2)

router.add\_connection('D', 'E', 1)

source = 'A'

destination = 'E'

path, distance = router.shortest\_path(source, destination)

print(f"Shortest path from {source} to {destination}: {' -> '.join(path)} with total cost {distance}")