4.22 NETWORK DELAY TIME USING DIJKSTRA’S ALGORITHM

Question:

You are given a network of n nodes, labeled from 1 to n. You are also given times, a list of travel times as directed edges times[i] = (ui, vi, wi), where ui is the source node, vi is the target node, and wi is the time it takes for a signal to travel from source to target. We will send a signal from a given node k. Return the minimum time it takes for all the n nodes to receive the signal. If it is impossible for all the n nodes to receive the signal, return -1.

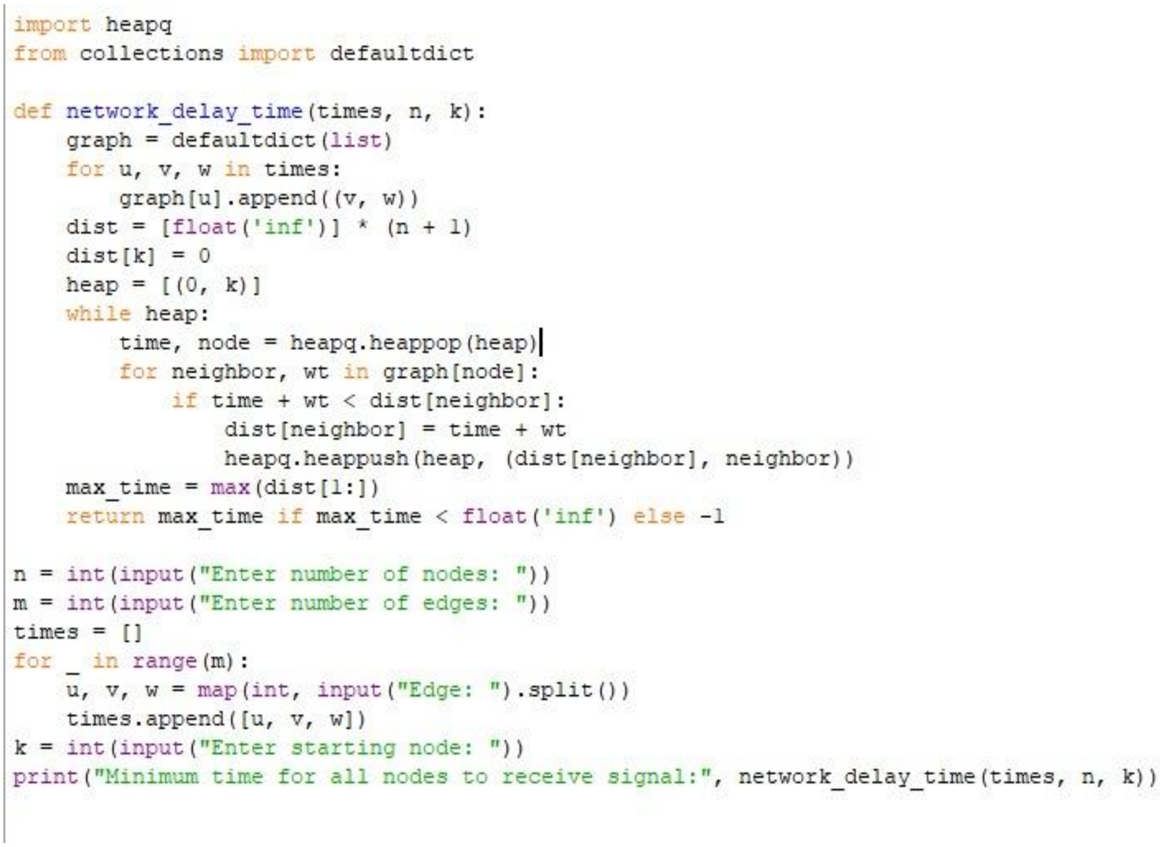
AIM

To compute the minimum time required for a signal to reach all nodes in a directed graph using Dijkstra’s algorithm.

ALGORITHM

1. Build an adjacency list graph from the times input.
2. Use a min-heap (priority queue) to always expand the node with the smallest known arrival time.
3. Initialize a dictionary dist to store the shortest time to each node.
4. Start from node k with time 0.
5. While the heap is not empty:
    - Pop the node with the smallest time.
    - For each neighbor, calculate the new time.
    - If the new time is better than the current one, update and push to the heap.
6. If all nodes are reached, return the maximum time in dist.
7. If any node is unreachable, return -1.

PROGRAM



```
import heapq
from collections import defaultdict

def network_delay_time(times, n, k):
    graph = defaultdict(list)
    for u, v, w in times:
        graph[u].append((v, w))
    dist = [float('inf')] * (n + 1)
    dist[k] = 0
    heap = [(0, k)]
    while heap:
        time, node = heapq.heappop(heap)
        for neighbor, wt in graph[node]:
            if time + wt < dist[neighbor]:
                dist[neighbor] = time + wt
                heapq.heappush(heap, (dist[neighbor], neighbor))
    max_time = max(dist[1:])
    return max_time if max_time < float('inf') else -1

n = int(input("Enter number of nodes: "))
m = int(input("Enter number of edges: "))
times = []
for _ in range(m):
    u, v, w = map(int, input("Edge: ").split())
    times.append([u, v, w])
k = int(input("Enter starting node: "))
print("Minimum time for all nodes to receive signal:", network_delay_time(times, n, k))
```

Input:

Enter number of nodes: 4

Enter number of edges: 3

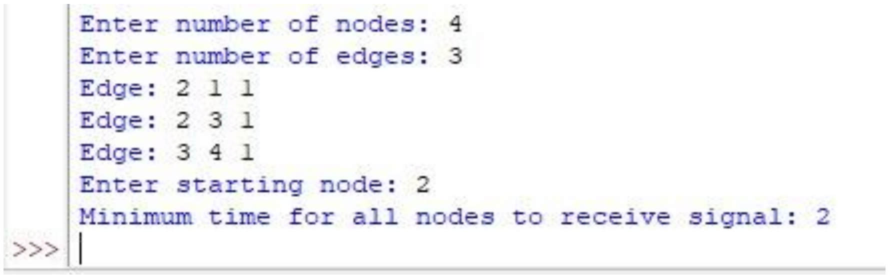
Edge: 2 1 1

Edge: 2 3 1

Edge: 3 4 1

Enter starting node: 2

Output:



```
Enter number of nodes: 4
Enter number of edges: 3
Edge: 2 1 1
Edge: 2 3 1
Edge: 3 4 1
Enter starting node: 2
Minimum time for all nodes to receive signal: 2
>>> |
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: O(E log N), where *E* is the number of edges and *N* is the number of nodes
- Space Complexity: O(N + E), for graph and heap.