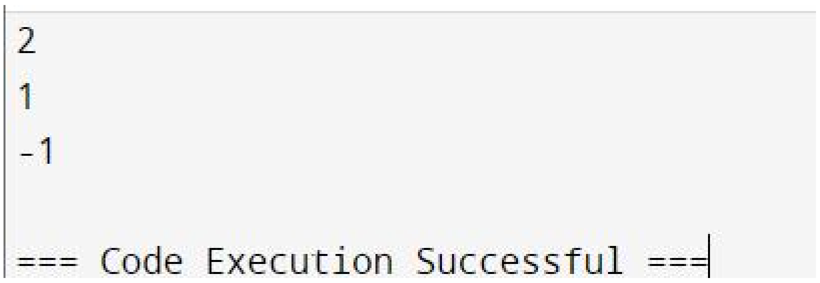
193. You are given a network of n nodes, labeled from 1 to n. You are also given times, a list of travel times as directed edges times[i] = (ui, vi, wi), where ui is the source node, vi is the target node, and wi is the time it takes for a signal to travel from source to target. We will send a signal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

PROGRAM:

```
import collections
import heapq

def network_delay_time(times, n, k):
    graph = collections.defaultdict(list)
    for u, v, w in times:
        graph[u].append((v, w))

    pq = [(0, k)]
    dist = {}

    while pq:
        time, node = heapq.heappop(pq)
        if node in dist:
            continue
        dist[node] = time
        for v, w in graph[node]:
            if v not in dist:
                heapq.heappush(pq, (time + w, v))

    return max(dist.values()) if len(dist) == n else -1

# Example 1
times = [[2,1,1],[2,3,1],[3,4,1]]
n = 4
k = 2
print(network_delay_time(times, n, k))  # Output: 2

# Example 2
times = [[1,2,1]]
n = 2
k = 1
print(network_delay_time(times, n, k))  # Output: 1

# Example 3
times = [[1,2,1]]
n = 2
k = 2
print(network_delay_time(times, n, k))  # Output: -1
```

OUTPUT:



```
2
1
-1

=== Code Execution Successful ===
```

TIME COMPLEXITY:O(E LOG E)