K부터 출발해 모든 노드가 신호를 받을 수 있는 시간을 계산하라. 불가능할 경우 -1을 리턴한다. 입력값 (u, v, w)는 각각 출발지, 도착지, 소요시간으로 구성되며, 전체 노드의 개수는 N으로 입력받는다.

Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2
Output: 2

1.dijkstra

```
import collections
import heapq

class Solution:
    def networkDelayTime(self, times: List[List[int]], n: int, k: int) -> int:
        graph = collections.defaultdict(list)
        for u, v, w in times:
            graph[u].append([v, w])
        dist = collections.defaultdict(int)
        for i in range(1, n + 1):
            if i == k:
                dist[i] = 0
            else:
                dist[i] = sys.maxsize
        q = [[k, 0]]
        while len(q) > 0:
            curr, wei = heapq.heappop(q)
            for u, w in graph[curr]:
                alt = wei + w
                if dist[u] > alt:
                    dist[u] = alt
                    heapq.heappush(q, [u, alt])
        for d in dist:
            if dist[d] == sys.maxsize:
                return -1
        return max(dist.values())

import collections
import heapq

class Solution:
    def networkDelayTime(self, times: List[List[int]], n: int, k: int) -> int:
        graph = collections.defaultdict(list)
        for u, v, w in times:
            graph[u].append((v,w))
        q = [(0, k)]
        dist = collections.defaultdict(int)
        while len(q) > 0:
            time, node = heapq.heappop(q)
            if node not in dist:
                dist[node] = time
                for v, w in graph[node]:
                    alt = time + w
            heapq.heappush(q, (alt, v))
if len(dist) == n:
    return max(dist.values())
return -1
```