**다익스트라**

1. **최단거리 ( #1753 )**

노드와 링크의 개수가 주어집니다.

출발한 노드의 번호가 주어집니다.

A,B로 가는 링크의 가중치 C가 주어집니다.

출발한 노드부터 각각의 노드에 도달하는 최단거리를 계산하세요.

**입력**

**5 6**

**1**

**5 1 1**

**1 2 2**

**1 3 3**

**2 3 4**

**2 4 5**

**3 4 6**

**출력**

**0**

**2**

**3**

**7**

**INF**

**정답코드**

**#1753**

**from** **collections** **import** deque

**import** **heapq**

N,M = map(int,input().split())

start = int(input())

links = [[] **for** \_ **in** range(N+1)]

dist = [ 1e9 **for** \_ **in** range(N+1) ]

*# 1,000,000,000*

**for** \_ **in** range(M):

A,B,C = map(int,input().split())

links[A].append([B,C])

*# bfs!*

q = []

heapq.heappush(q,[0,start])

dist[start] = 0

**while** q: *# q 배열에 아무것도 없으면 False가 됩니다.*

*# dist를 보고 순서를 바꾸는 코드!*

\_w,node = heapq.heappop(q)

**for** nxt, weight **in** links[node]:

*# 1. 각각의 노드에 값을 업데이트*

*# 2. 만약에 여러 경로가 있다면 min 비교!*

*# 3. 시작점으로부터 거리를 봐서, 짧은 순서대로 탐색! ( 오염 ! )*

**if** dist[node] + weight < dist[nxt]:

dist[nxt] = dist[node] + weight

heapq.heappush(q,[dist[nxt],nxt])

**for** j **in** range(1,N+1):

**if** dist[j] == 1e9:

print(*"INF"*)

**else**:

print(dist[j])