# 문제

N개의 도시가 있다. 그리고 한 도시에서 출발하여 다른 도시에 도착하는 버스가 M개 있다. 각 버스는 A, B, C로 나타낼 수 있는데, A는 시작도시, B는 도착도시, C는 버스를 타고 이동하는데 걸리는 시간이다. 시간 C가 양수가 아닌 경우가 있다. C = 0인 경우는 순간 이동을 하는 경우, C < 0인 경우는 타임머신으로 시간을 되돌아가는 경우이다.

1번 도시에서 출발해서 나머지 도시로 가는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 도시의 개수 N (1 ≤ N ≤ 500), 버스 노선의 개수 M (1 ≤ M ≤ 6,000)이 주어진다. 둘째 줄부터 M개의 줄에는 버스 노선의 정보 A, B, C (1 ≤ A, B ≤ N, -10,000 ≤ C ≤ 10,000)가 주어진다.

출력

만약 1번 도시에서 출발해 어떤 도시로 가는 과정에서 시간을 무한히 오래 전으로 되돌릴 수 있다면 첫째 줄에 -1을 출력한다. 그렇지 않다면 N-1개 줄에 걸쳐 각 줄에 1번 도시에서 출발해 2번 도시, 3번 도시, ..., N번 도시로 가는 가장 빠른 시간을 순서대로 출력한다. 만약 해당 도시로 가는 경로가 없다면 대신 -1을 출력한다.

예제 입력 1 복사

3 4

1 2 4

1 3 3

2 3 -1

3 1 -2

예제 출력 1 복사

4

3

예제 입력 2 복사

3 4

1 2 4

1 3 3

2 3 -4

3 1 -2

예제 출력 2 복사

-1

예제 입력 3 복사

3 2

1 2 4

1 2 3

예제 출력 3 복사

3

-1

# 풀이

[문제를 해결한 방법을 자유롭게 작성]

벨먼포드 알고리즘은 음의 가중치를 가진 간선을 포함하는 그래프에서도 동작한다. 해당 문제 조건을 보면 버스 이동 시간이 음수가 될 수 있기에 벨먼 포드 알고리즘을 이용해서 풀었다.

벨먼포드와 비슷한 알고리즘인 ‘다익스트리 알고리즘’은 같은 최단 경로 찾기 알고리즘이지만. 가중치가 음수가 될 수 없다.

[소스코드

# 무한대를 나타내는 값

INF = float('inf')

# n m을 입력받는다.

n, m = map(int, sys.stdin.readline().split())

bus = []

#노선 정보를 a b c를 입력 받는다.

for \_ in range(m):

a, b, c = map(int, sys.stdin.readline().split())

bus.append((a, b, c))

# 최단 거리를 저장할 리스트 dist를 무한대로 초기화한다.

#(아직 어떤 도시로 가는 경로도 알지 못하기 때문)

dist = [INF]\*(n+1)

dist[1] = 0 # 시작 도시인 1번 도시까지의 거리는 0이다.

# 벨만-포드 알고리즘

def bf(start):

dist[start] = 0 # 시작 도시까지의 거리는 0

# 모든 도시에 대해 n번 반복

# n번 반복하는 이유는 최악의 경우 모든 도시를 순회하는 경로를 고려하기 때문

for i in range(n):

# 모든 버스 노선에 대해 반복한다.

for j in range(m):

cur = bus[j][0] # 현재 도시

next\_node = bus[j][1] # 다음 도시

cost = bus[j][2] # 현재 도시에서 다음 도시로 가는데 필요한 시간

#최단 거리가 '무한대'가 아니고, 다음 도시까지의 거리가 '현재 도시까지의 거리 + 현재 도시에서 다음 도시로 가는데 필요한 시간 보다 크면 값을 갱신합니다.

if dist[cur] != INF and dist[next\_node] > dist[cur] + cost:

dist[next\_node] = dist[cur] + cost

# n번째 라운드에서도 값이 갱신된다면 음수 순환이 존재

if i == n - 1:

return True

return False

# 벨만-포드 알고리즘을 실행하여 음수를 확인하는 문장

negative\_cycle = bf(1)

# 음수가 있으면ㄴ -1을 출력

if negative\_cycle:

print(-1)

else:

# 1번 도시에서 출발해 2번 도시, 3번 도시, N번 도시로 가는 가장 빠른 시간을 출력한다.

for i in range(2, n+1):

# 해당 도시로 가는 경로가 없다면 대신 -1을 출력합니다.

if dist[i] == INF:

print(-1)

else:

print(dist[i])

[결과화면]



