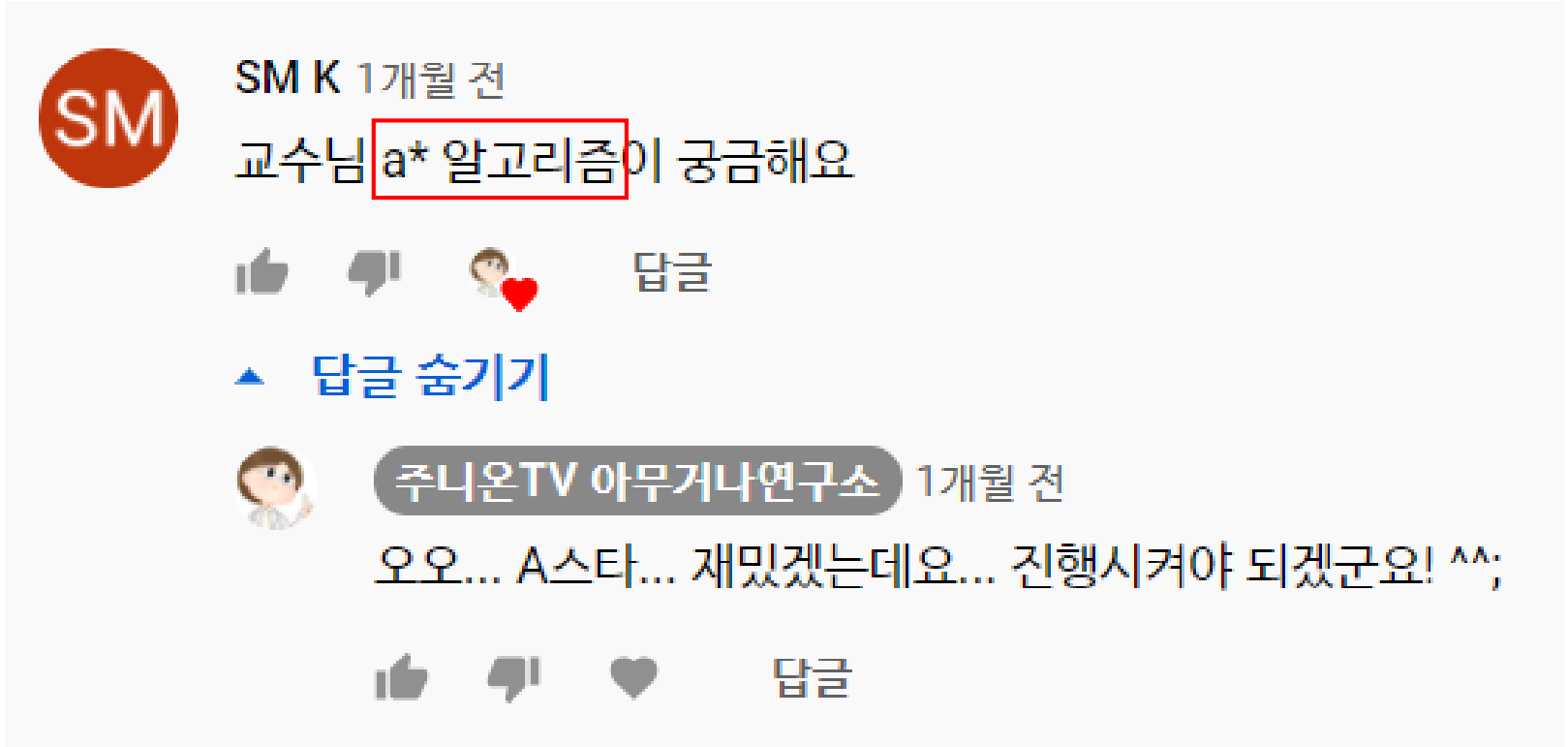
아무거나 물어보살

오늘의 질문:

A* 알고리즘이 궁금해요!

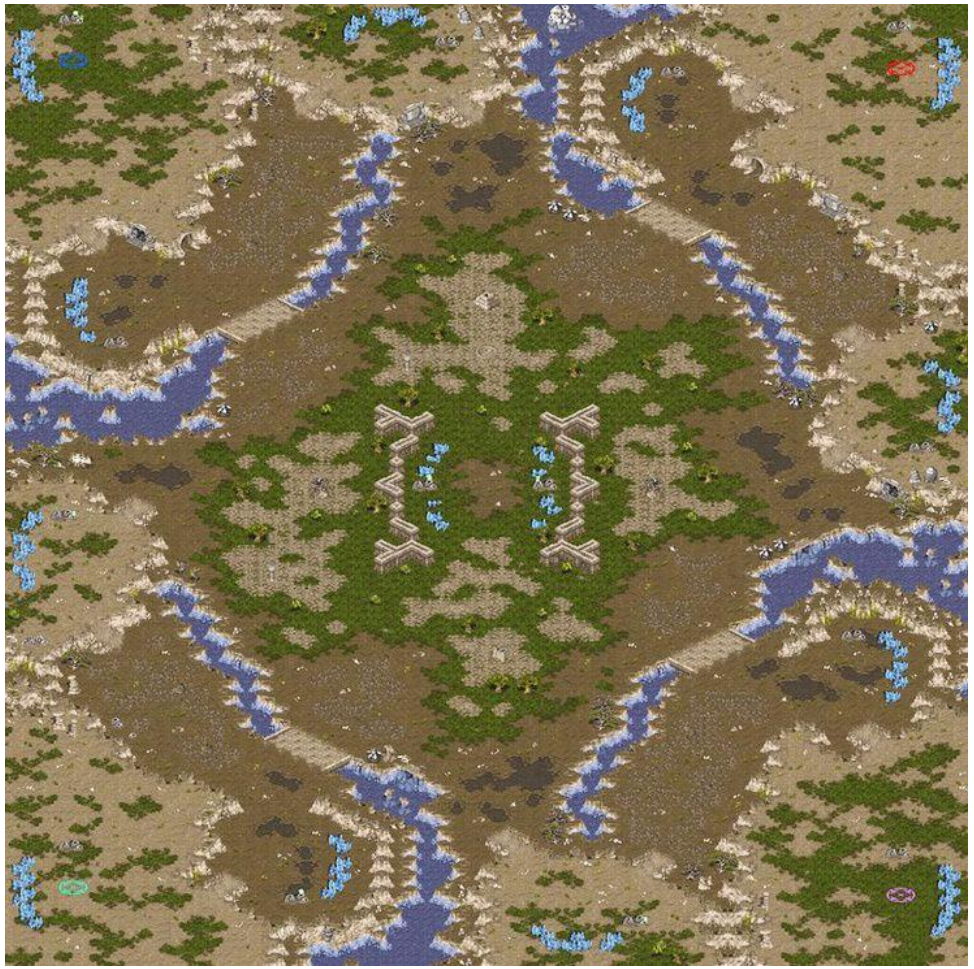
아무거나 물어보살: A* 알고리즘이 궁금해요!

- 오늘의 질문:

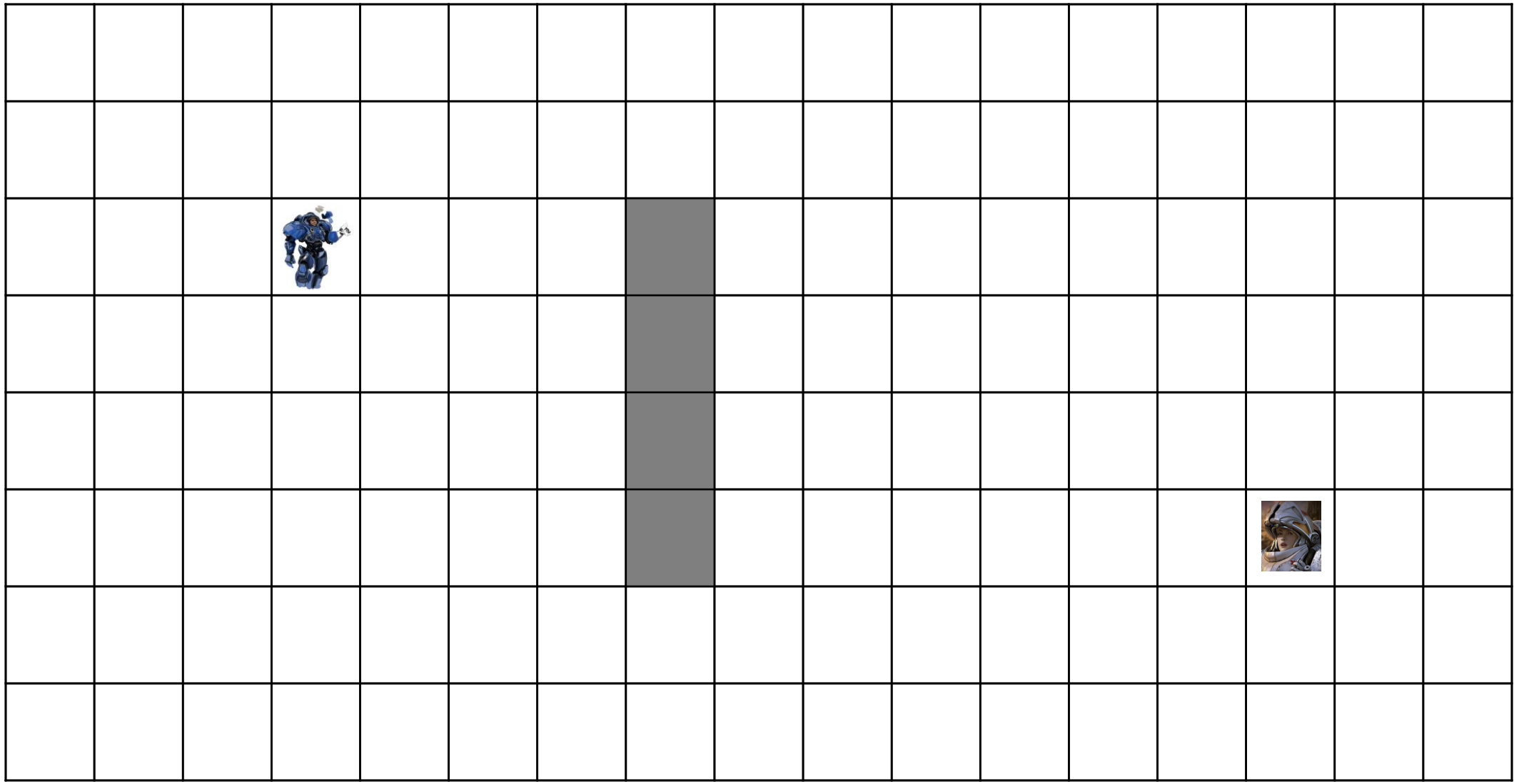


아무거나 물어보살: A* 알고리즘이 궁금해요!

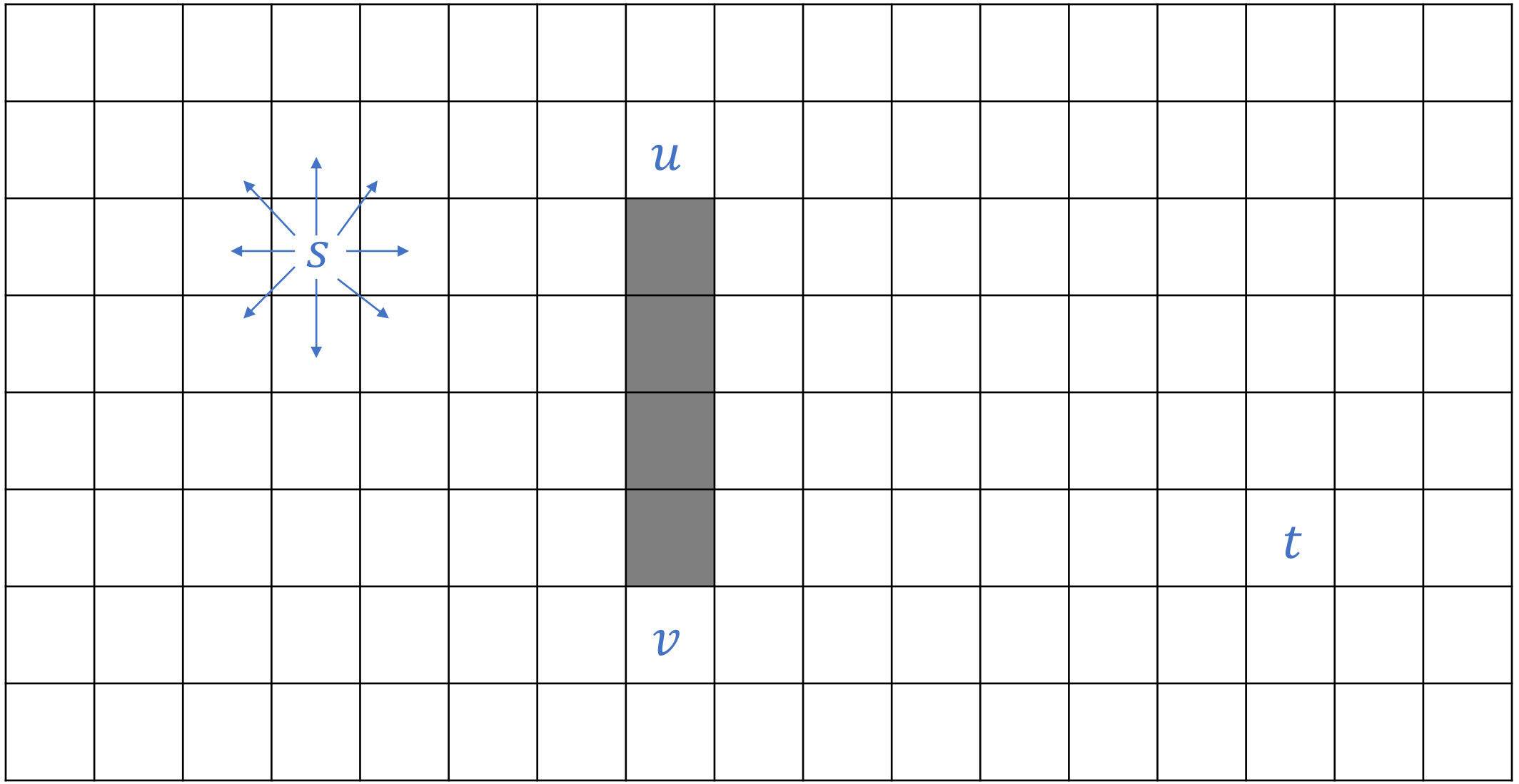
- 최단 경로 문제와 A* 알고리즘:



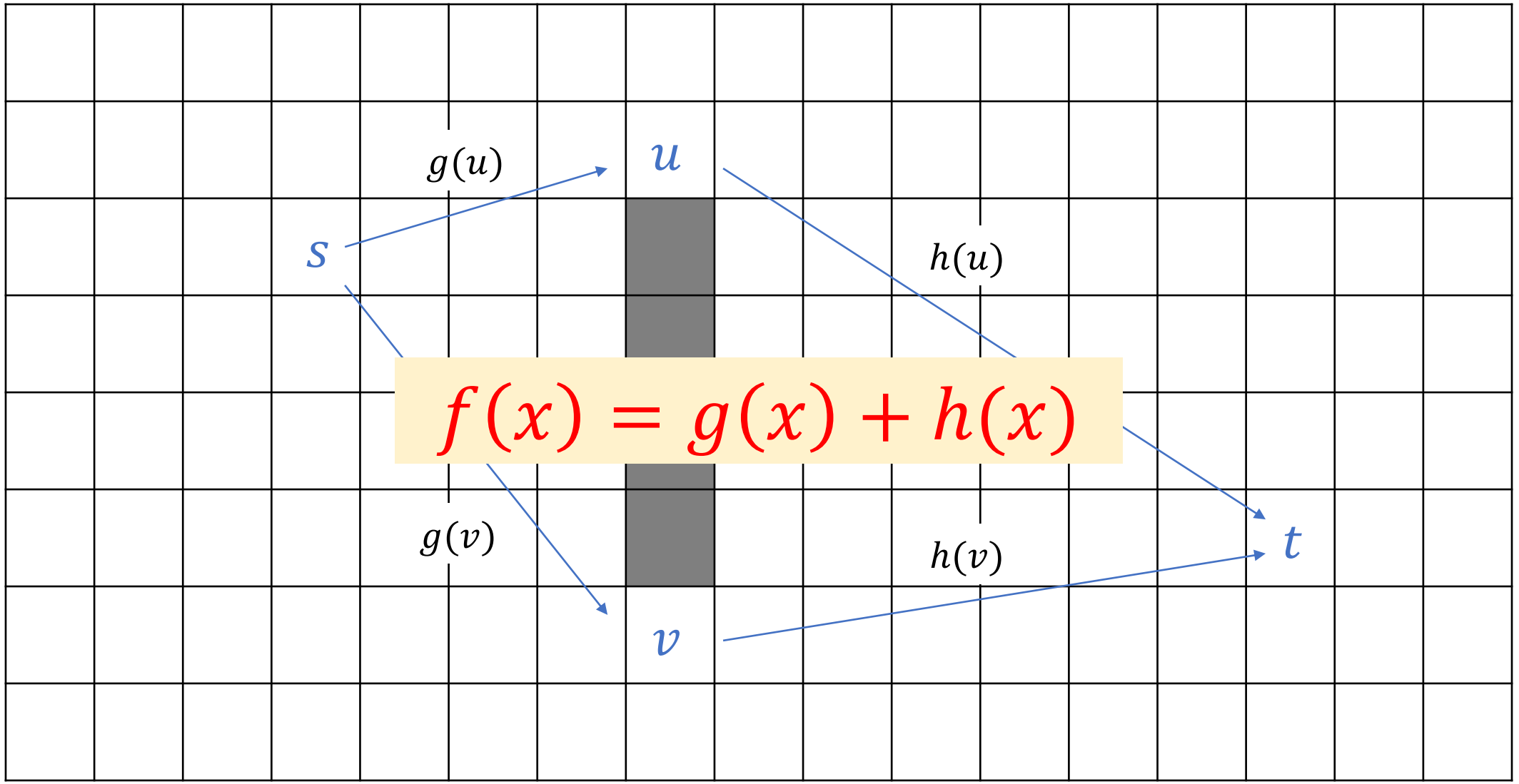
아무거나 물어보살: A* 알고리즘이 궁금해요!



아무거나 물어보살: A* 알고리즘이 궁금해요!



아무거나 물어보살: A* 알고리즘이 궁금해요!

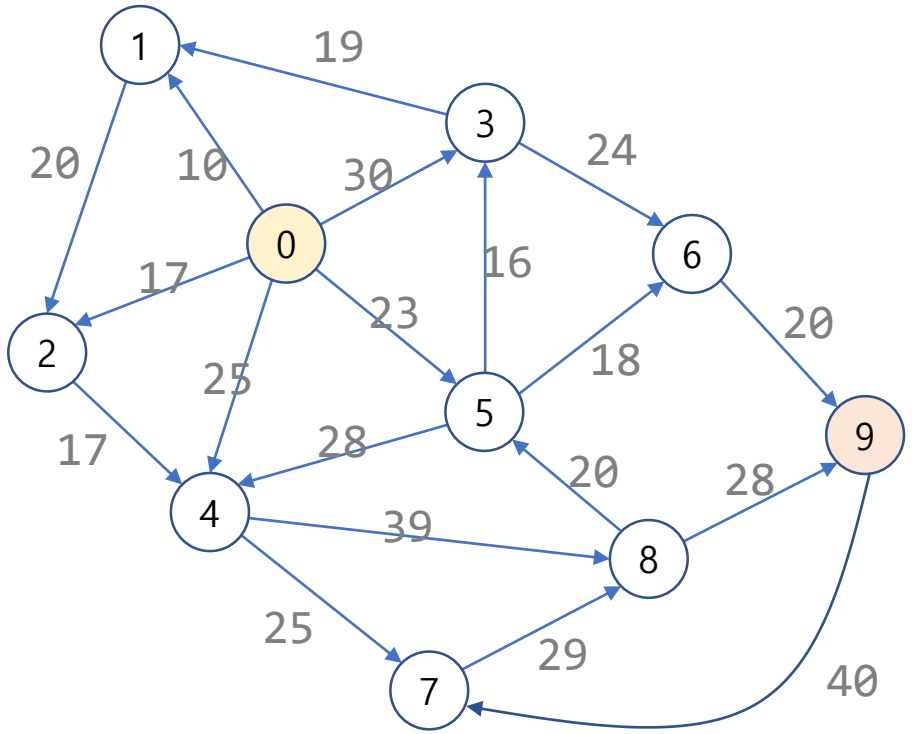


아무거나 물어보살: A* 알고리즘이 궁금해요!

- **최단 경로 알고리즘:**
  - 플로이드 알고리즘: *Dynamic Programming*, $O(n^3)$
    - 모든 쌍(모든 정점에서 다른 모든 정점으로)의 최단 경로 구하기
  - 다익스트라 알고리즘: *Greedy Approach*, $O(n^2)$
    - 단일 출발점에서 다른 모든 정점으로의 최단 경로 구하기
  - A* 알고리즘: *Heuristic Algorithm*
    - 출발점과 도착점이 모두 주어졌을 때 최단 경로 구하기

아무거나 물어보살: A* 알고리즘이 궁금해요!

- 예제: 가중치 있는 방향 그래프 $G = (V, E)$



graph.1.in:

```
10 19
0 1 10
0 2 17
0 3 30
0 4 25
0 5 23
1 2 20
2 4 17
3 1 19
3 6 24
4 7 25
4 8 39
5 3 16
5 4 28
5 6 18
6 9 20
7 8 29
8 5 20
8 9 28
9 7 40
```

- 그래프 $G = (V, E)$로부터 인접 행렬 W 만들기:

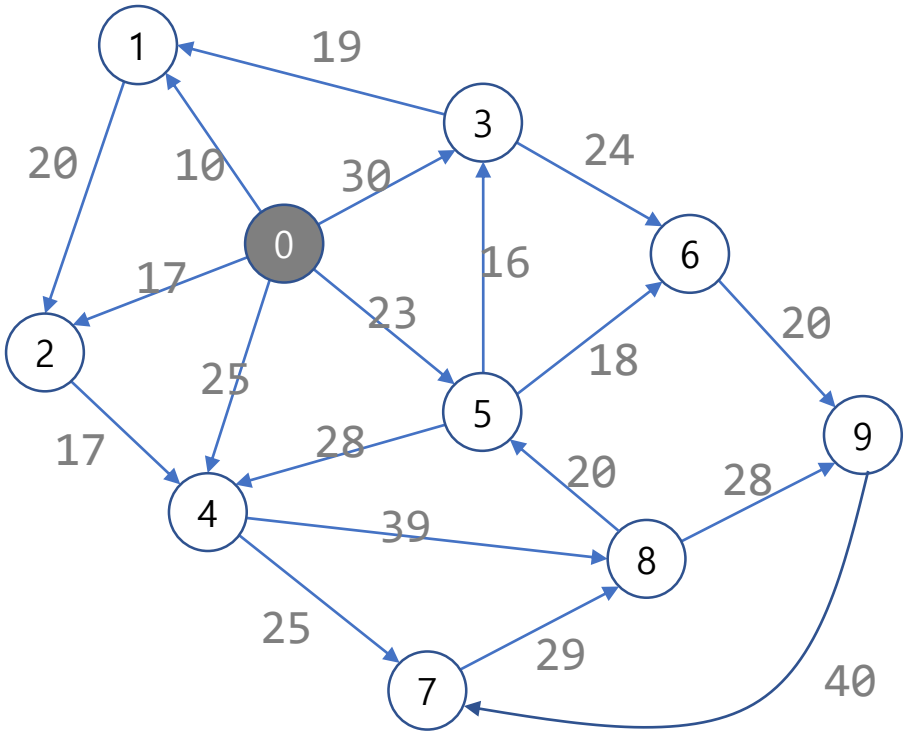
```
INF = 0xFFFF
n, m = map(int, input().split())
W = [[0 if i == j else INF for i in range(n)] for j in range(n)]
for _ in range(m):
    u, v, w = map(int, input().split())
    W[u][v] = w

s, t = 0, 9
bypass = dijkstra(s, n, W)
path = shortest(s, t, bypass)
print(bypass)
print(path)
```

아무거나 물어보살: A* 알고리즘이 궁금해요!

- 다익스트라 알고리즘: 출발 정점 s가 주어질 때.
  - $length[v]$: s로부터 v에 도달하는 최단 경로 길이
  - $bypass[v]$: s로부터 v에 도달하기 위해 거쳐가는 정점

```
def dijkstra(s, n, W):
    bypass = [s for _ in range(n)]
    length = [W[s][i] for i in range(n)]
    for _ in range(n - 1):
        vnear = nearest(s, length)
        for v in range(n):
            if length[v] > length[vnear] + W[vnear][v]:
                length[v] = length[vnear] + W[vnear][v]
                bypass[v] = vnear
        length[vnear] = -1
    return bypass
```

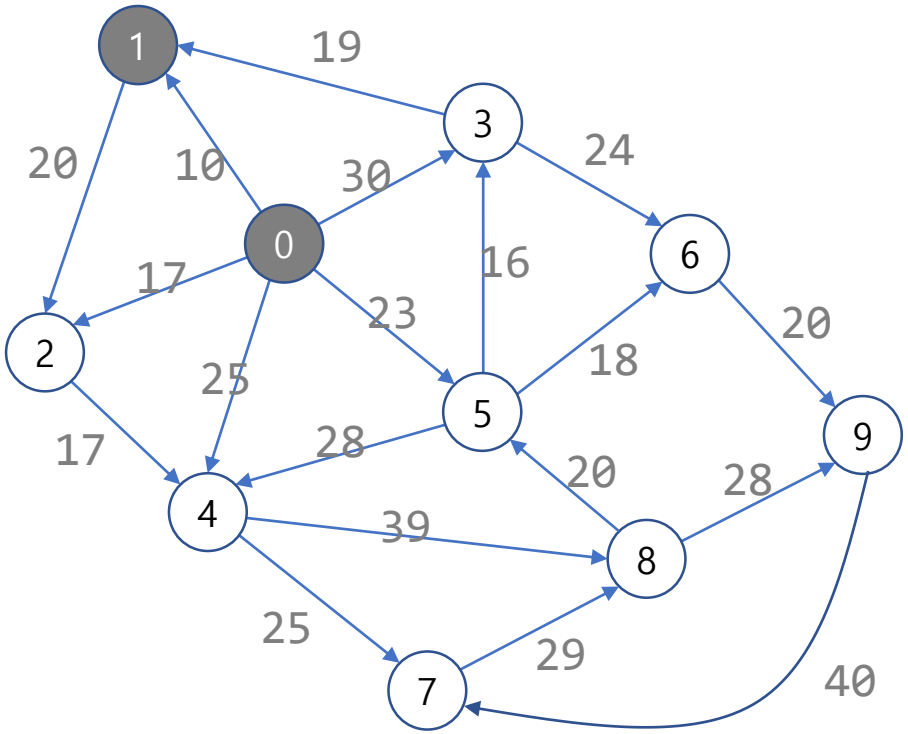
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	10	17	30	25	23	∞	∞	∞	∞
bypass	0	0	0	0	0	0	0	0	0	0

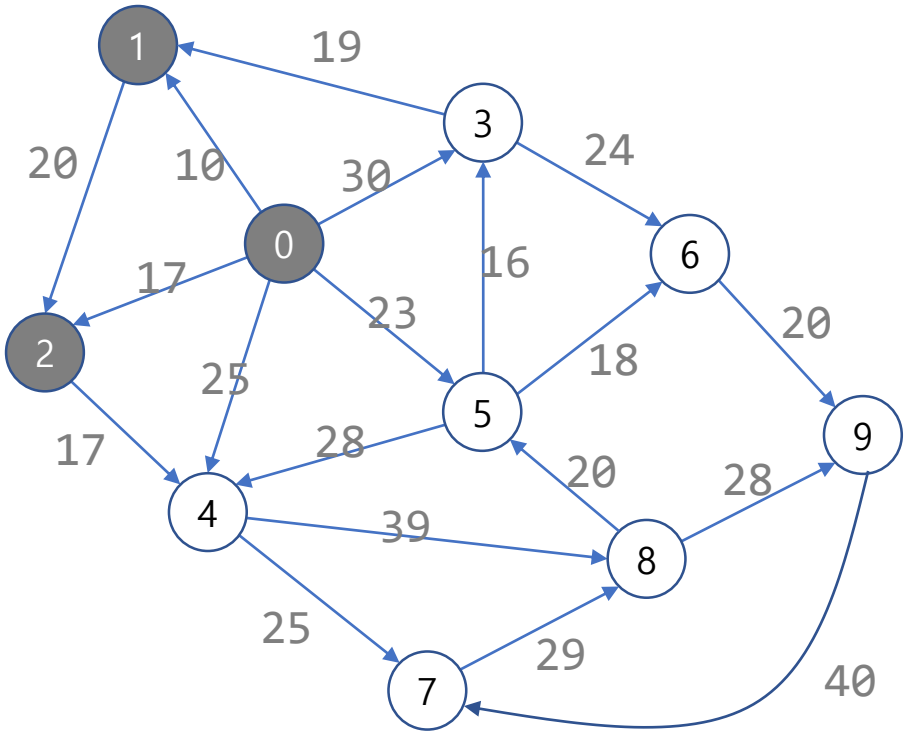
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	17	30	25	23	∞	∞	∞	∞
bypass	0	0	0	0	0	0	0	0	0	0

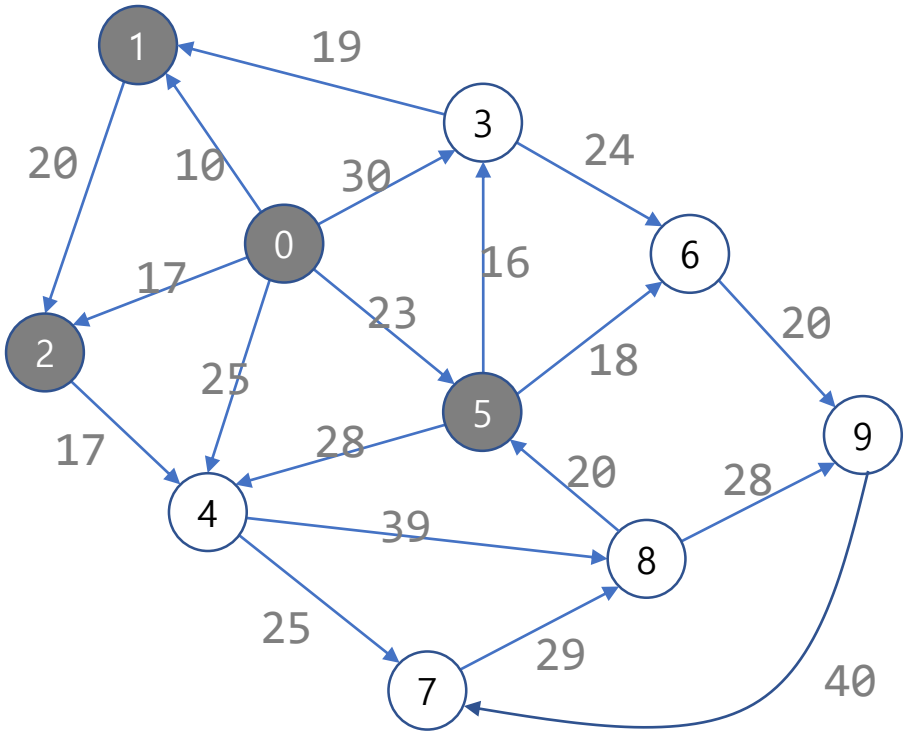
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	30	25	23	∞	∞	∞	∞
bypass	0	0	0	0	0	0	0	0	0	0

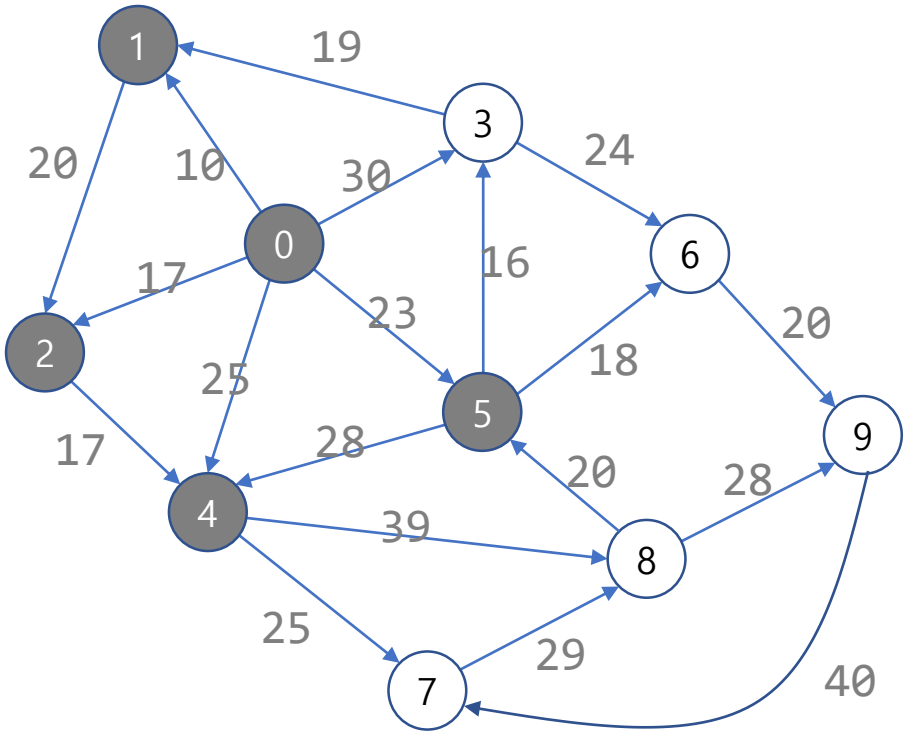
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	30	25	-1	41	∞	∞	∞
bypass	0	0	0	0	0	0	5	0	0	0

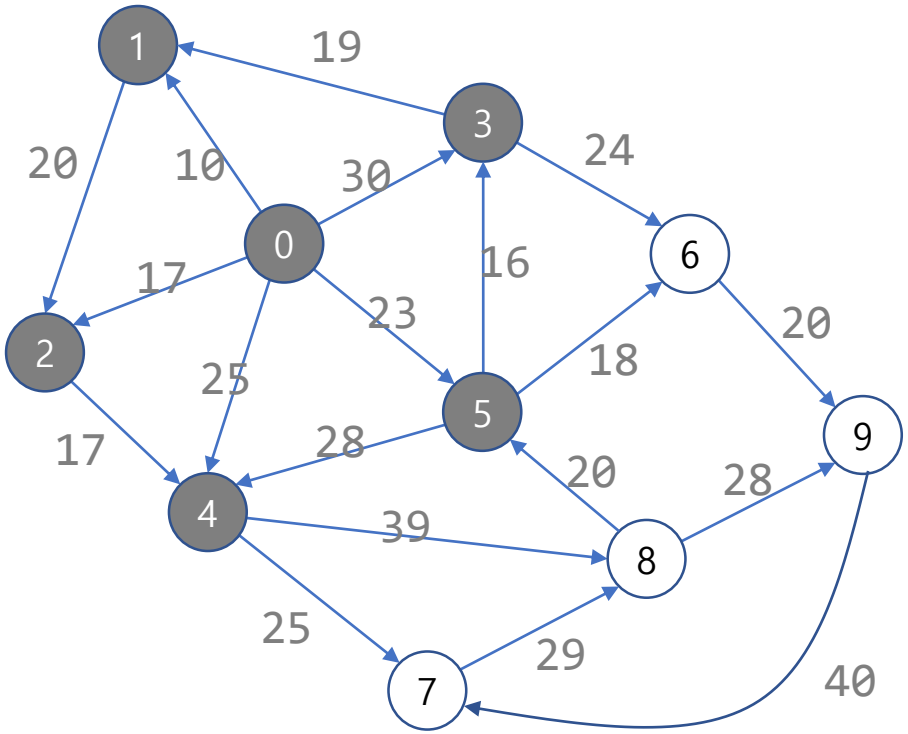
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5, 4\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	30	-1	-1	41	50	64	∞
bypass	0	0	0	0	0	0	5	4	4	0

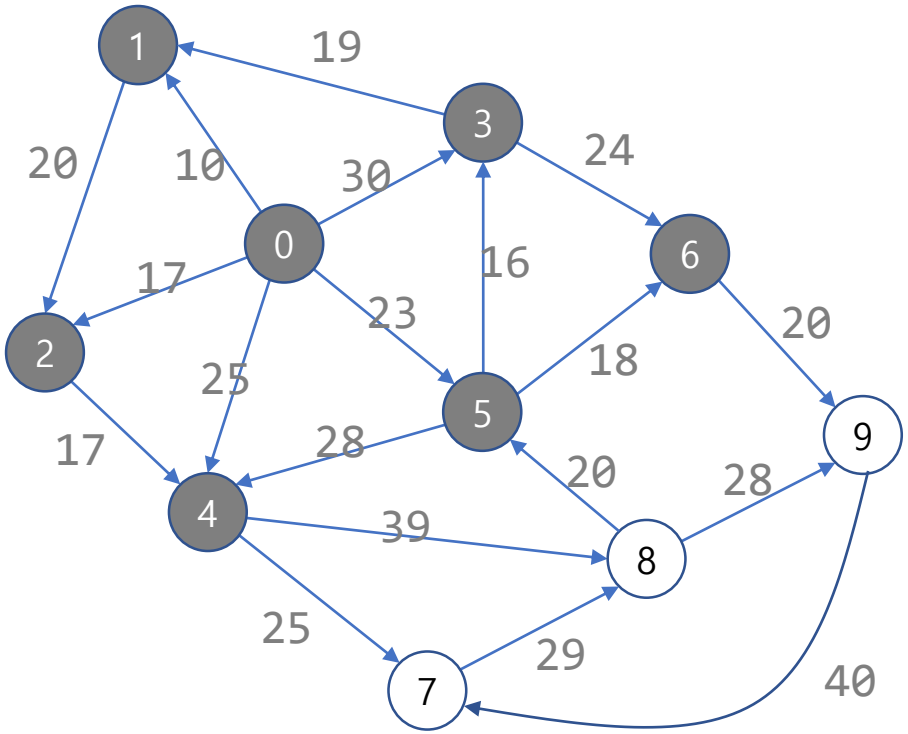
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5, 4, 3\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	-1	-1	-1	41	50	64	∞
bypass	0	0	0	0	0	0	5	4	4	0

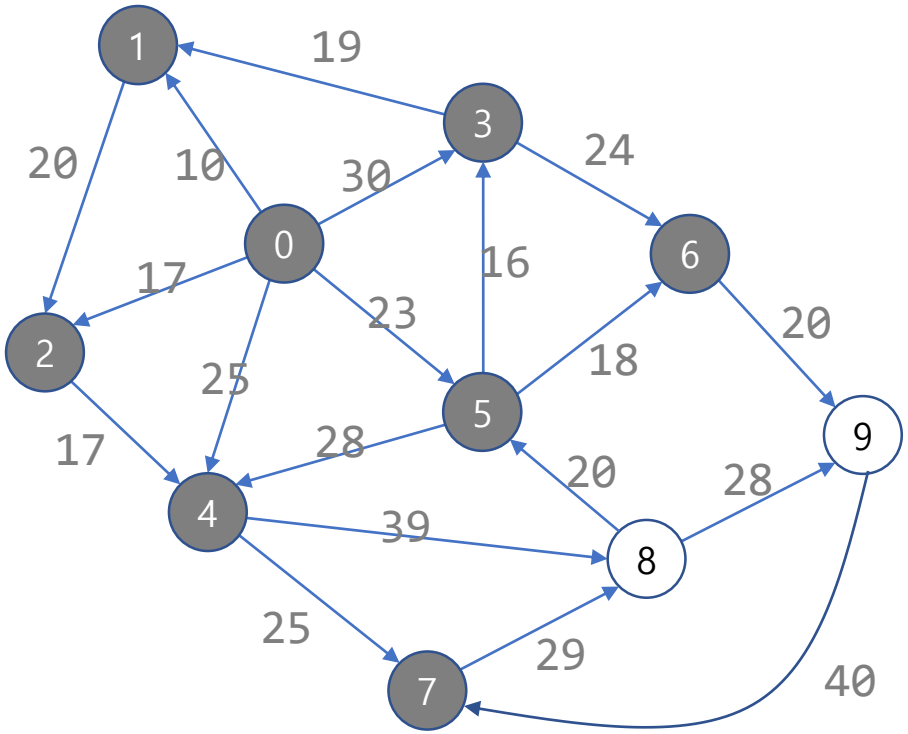
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5, 4, 3, 6\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	-1	-1	-1	-1	50	64	61
bypass	0	0	0	0	0	0	5	4	4	6

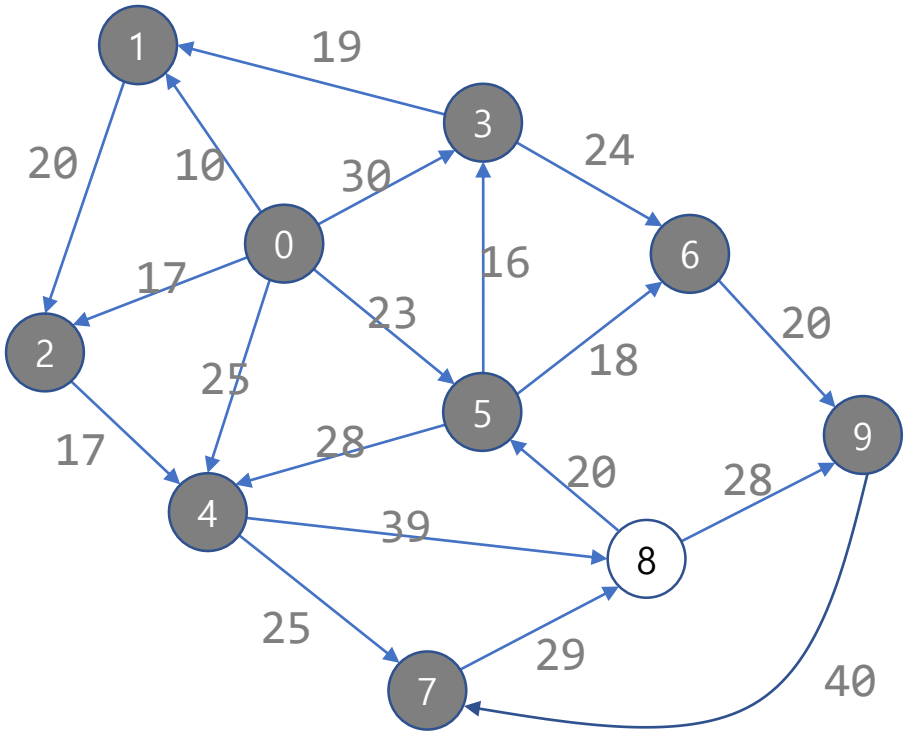
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5, 4, 3, 6, 7\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	-1	-1	-1	-1	-1	64	61
bypass	0	0	0	0	0	0	5	4	4	6

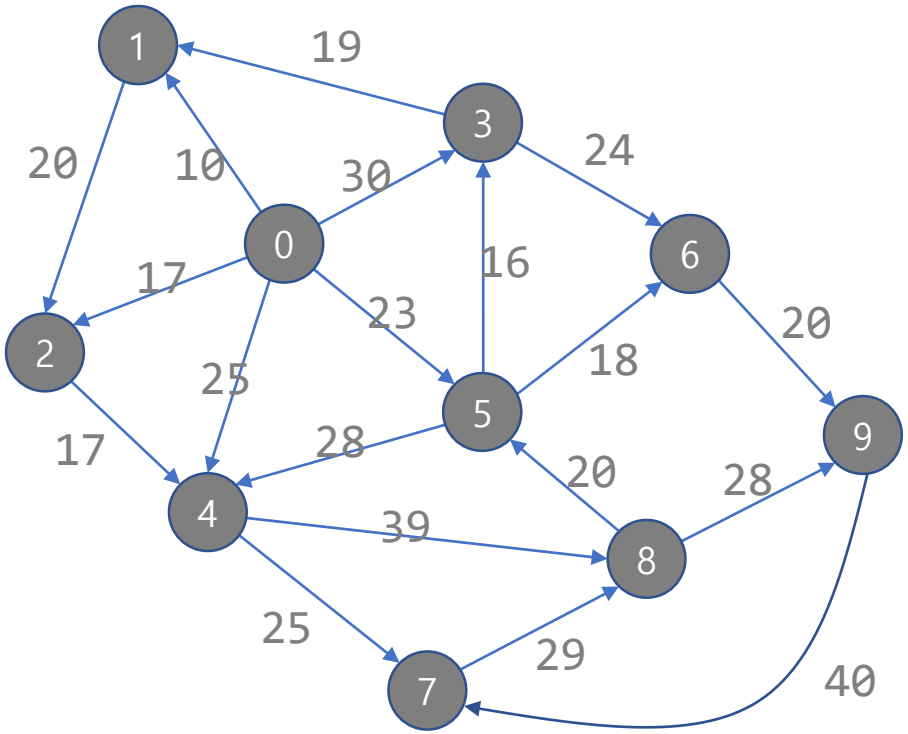
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5, 4, 3, 6, 7, 9\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	-1	-1	-1	-1	-1	64	-1
bypass	0	0	0	0	0	0	5	4	4	6

아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{0, 1, 2, 5, 4, 3, 6, 7, 9, 8\}$

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
bypass	0	0	0	0	0	0	5	4	4	6

아무거나 물어보살: A* 알고리즘이 궁금해요!

```
def nearest(s, length):
    minimum, vnear = INF, s
    for v in range(len(length)):
        if v == s: continue
        if 0 <= length[v] < minimum:
            minimum, vnear = length[v], v
    return vnear
```

v	0	1	2	3	4	5	6	7	8	9
length	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
bypass	0	0	0	0	0	0	5	4	4	6

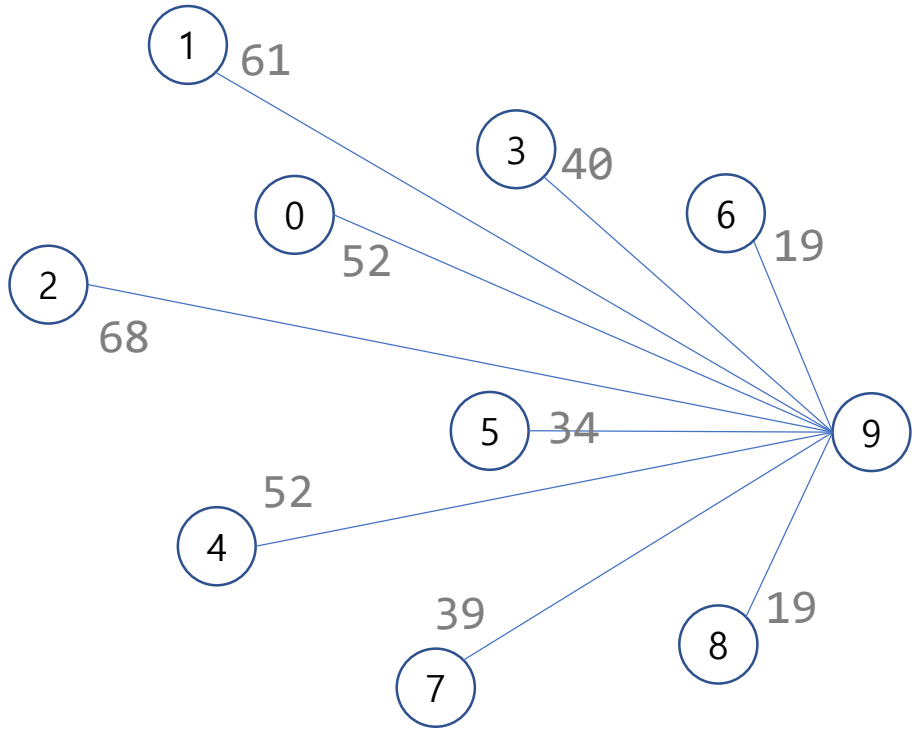
```
def shortest(s, t, bypass):
    if bypass[t] == s:
        return [s, t]
    else:
        return shortest(s, bypass[t], bypass) + [t]
```

아무거나 물어보살: A* 알고리즘이 궁금해요!

- A* 알고리즘: 출발 정점 s와 도착 정점 t가 주어질 때.
  - 휴리스틱: 각 정점 간의 추정 거리를 알 수 있다고 가정함
  - $g(v)$: 출발 정점 s에서 임의의 정점 v에 이르는 최단 거리
  - $h(v)$: 임의의 정점 v에서 도착 정점 t에 이르는 **추정 거리**
  - $f(v) = g(v) + h(v)$

graph.2.in:

```
// graph.in.in
// ……
9
0 9 52
1 9 61
2 9 68
3 9 40
4 9 52
5 9 34
6 9 19
7 9 39
8 9 19
```



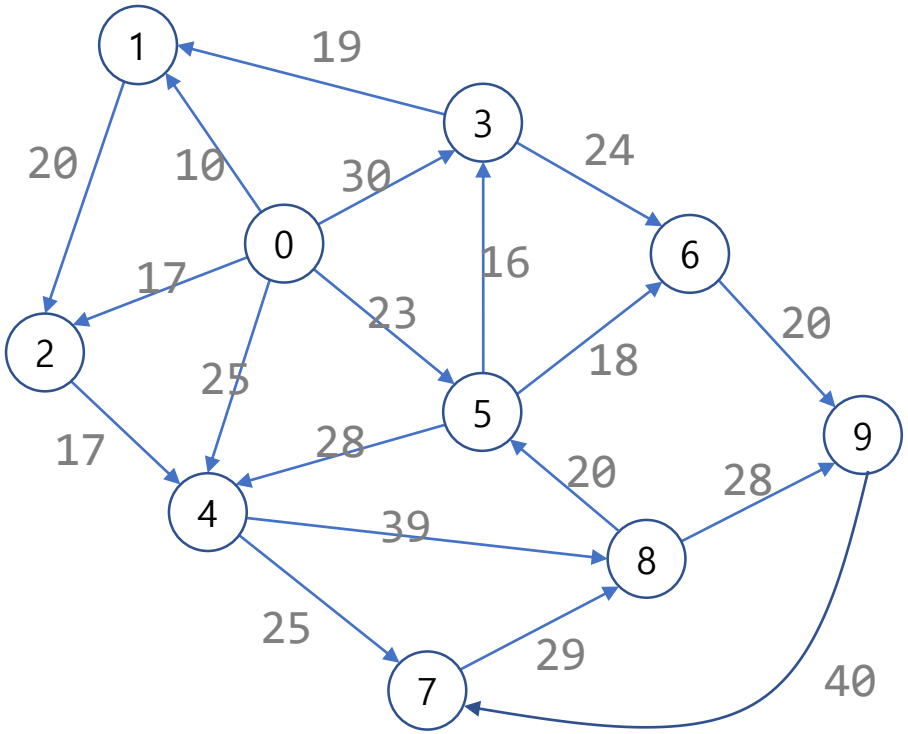
```
INF = 0xFFFF
n, m = map(int, input().split())
W = [[0 if i == j else INF for i in range(n)] for j in range(n)]
for _ in range(m):
    u, v, w = map(int, input().split())
    W[u][v] = w
H = [[0 if i == j else INF for i in range(n)] for j in range(n)]
k = int(input())
for _ in range(k):
    u, v, h = map(int, input().split())
    H[u][v] = h
```

아무거나 물어보살: A* 알고리즘이 궁금해요!

- A* 알고리즘의 구현:

```
def a_star(s, t, n, W, H):
    bypass = [s for _ in range(n)]
    g = [0 if i == s else INF for i in range(n)]
    h = [H[i][t] for i in range(n)]
    f = [h[s] if i == s else INF for i in range(n)]
    for _ in range(n - 1):
        vnear = nearest(s, f)
        if vnear == t:
            return bypass
        for v in range(n):
            if g[v] > g[vnear] + W[vnear][v]:
                g[v] = g[vnear] + W[vnear][v]
                f[v] = g[v] + h[v]
                bypass[v] = vnear
        f[vnear] = -1
    return bypass
```

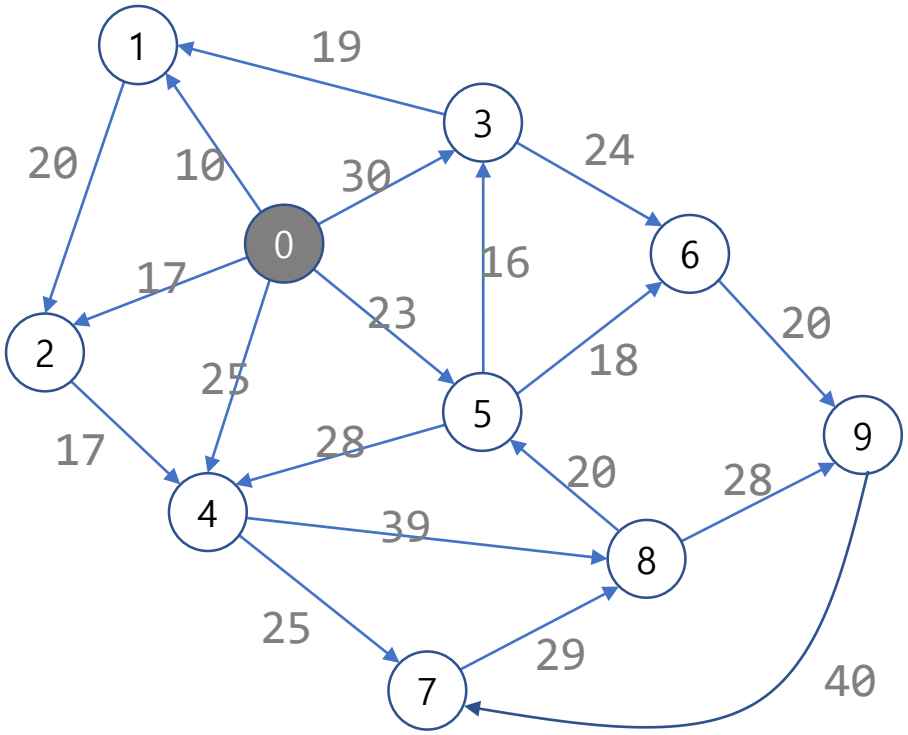
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{\ \}$

v	0	1	2	3	4	5	6	7	8	9
h	52	61	68	40	52	34	19	39	19	0
g	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
f	52	∞	∞	∞	∞	∞	∞	∞	∞	∞
bypass	0	0	0	0	0	0	0	0	0	0

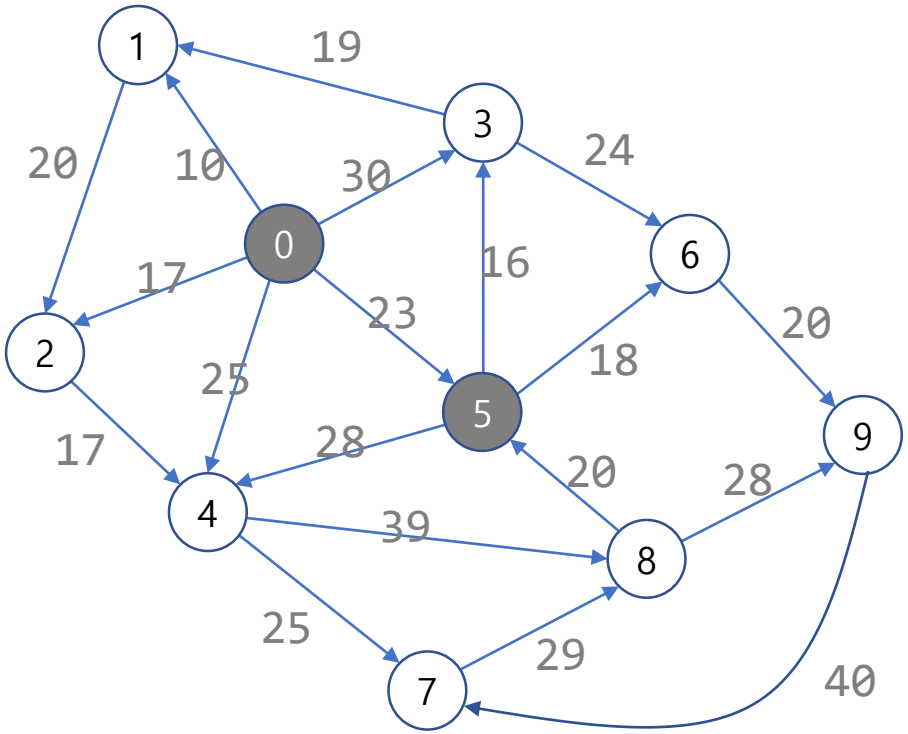
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{ 0 \}$

v	0	1	2	3	4	5	6	7	8	9
h	52	61	68	40	52	34	19	39	19	0
g	0	10	17	30	25	23	∞	∞	∞	∞
f	-1	71	85	70	77	57	∞	∞	∞	∞
bypass	0	0	0	0	0	0	0	0	0	0

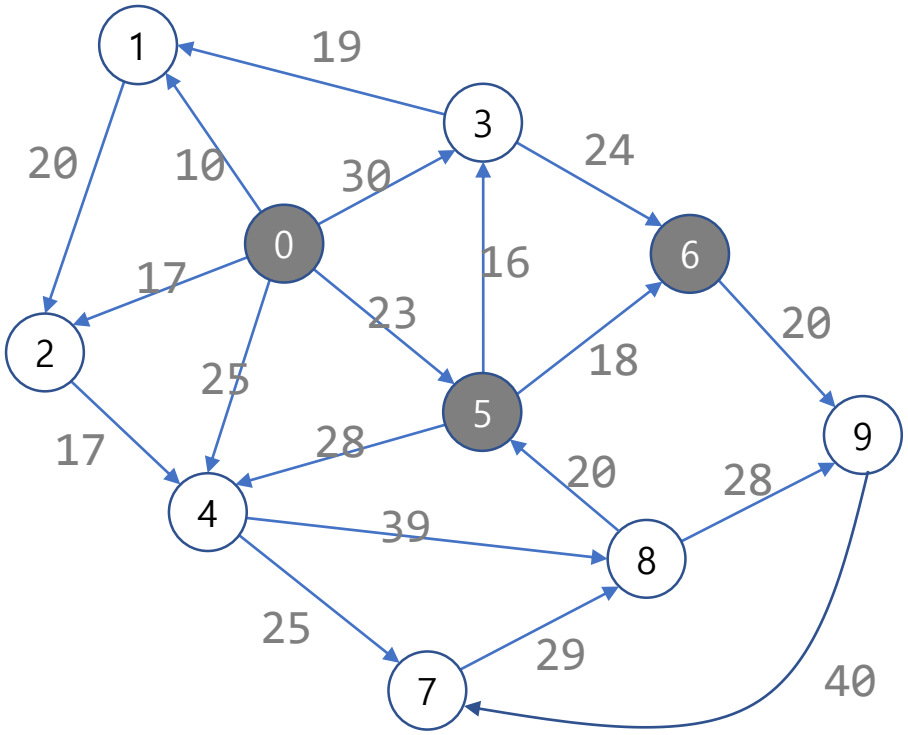
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{ 0, 5 \}$

v	0	1	2	3	4	5	6	7	8	9
h	52	61	68	40	52	34	19	39	19	0
g	0	10	17	30	25	23	41	∞	∞	∞
f	-1	71	85	70	77	-1	60	∞	∞	∞
bypass	0	0	0	0	0	0	5	0	0	0

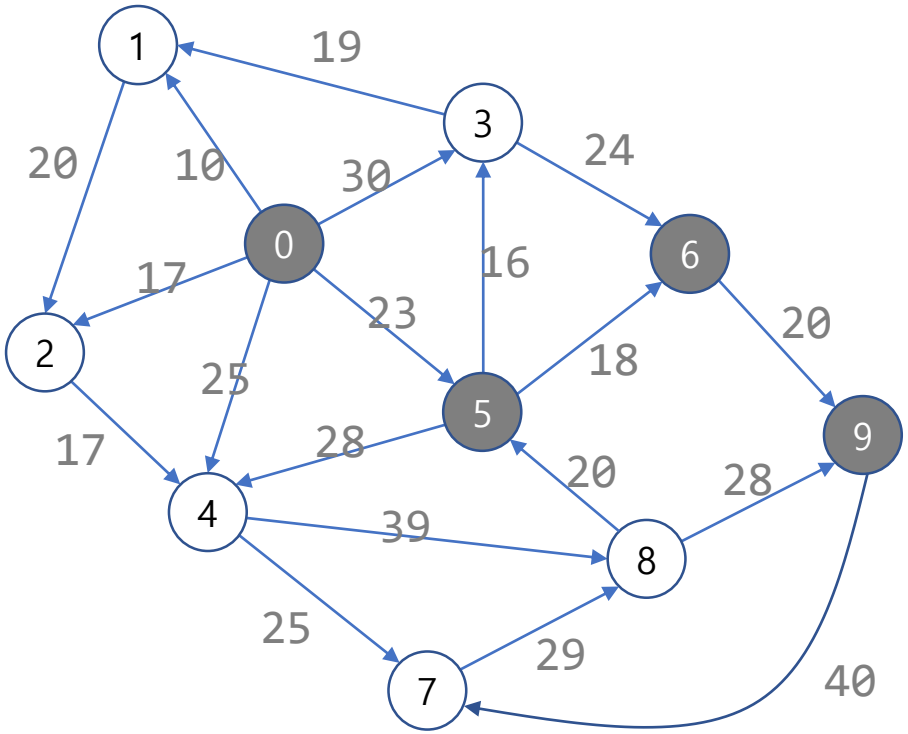
아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{ 0, 5, 6 \}$

v	0	1	2	3	4	5	6	7	8	9
h	52	61	68	40	52	34	19	39	19	0
g	0	10	17	30	25	23	41	∞	∞	61
f	-1	71	85	70	77	-1	-1	∞	∞	61
bypass	0	0	0	0	0	0	5	0	0	6

아무거나 물어보살: A* 알고리즘이 궁금해요!



$V = \{ 0, 5, 6, 9 \}$

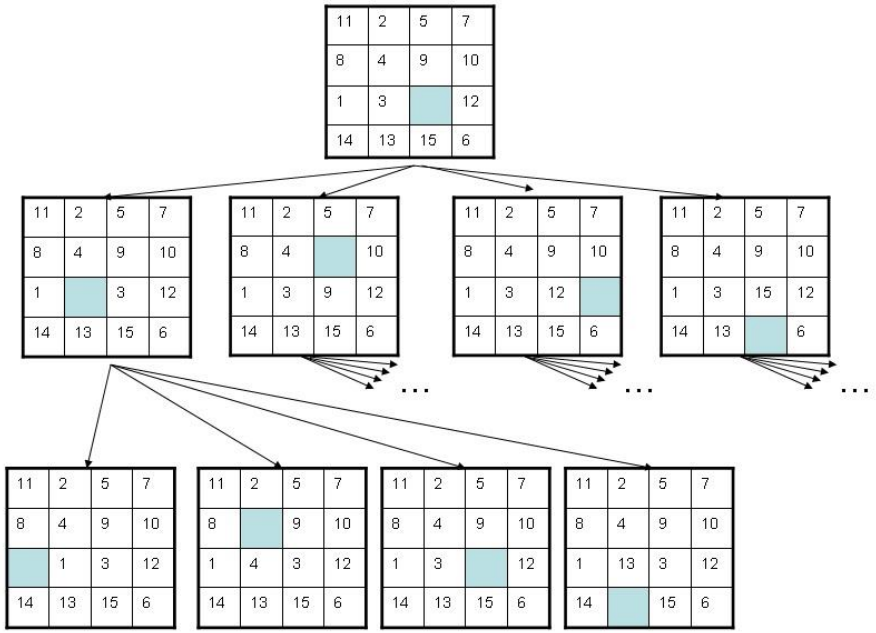
v	0	1	2	3	4	5	6	7	8	9
h	52	61	68	40	52	34	19	39	19	0
g	0	10	17	30	25	23	41	∞	∞	61
f	-1	71	85	70	77	-1	-1	∞	∞	-1
bypass	0	0	0	0	0	0	5	0	0	6

아무거나 물어보살: A* 알고리즘이 궁금해요!

```
s, t = 0, 9
bypass = a_star(s, t, n, W, H)
path = shortest(s, t, bypass)
print(bypass)
print(path)
```

아무거나 물어보살: A* 알고리즘이 궁금해요!

- A* 알고리즘:
  - 출발지와 도착지가 주어질 때 최단 경로를 찾는 휴리스틱 알고리즘
  - 발견한 최단 경로와 남은 추정 거리의 합이 최소가 되는 경로를 선택
  - 남은 추정 거리 계산: 유클리드 거리, 맨하탄 거리, 코사인 유사도 등
  - 응용 분야: 컴퓨터 게임, 자동차 네비게이션, 15-퍼즐 문제 등




Any Questions?
주니온TV@Youtube
자세히 보면 유익한 코딩 채널