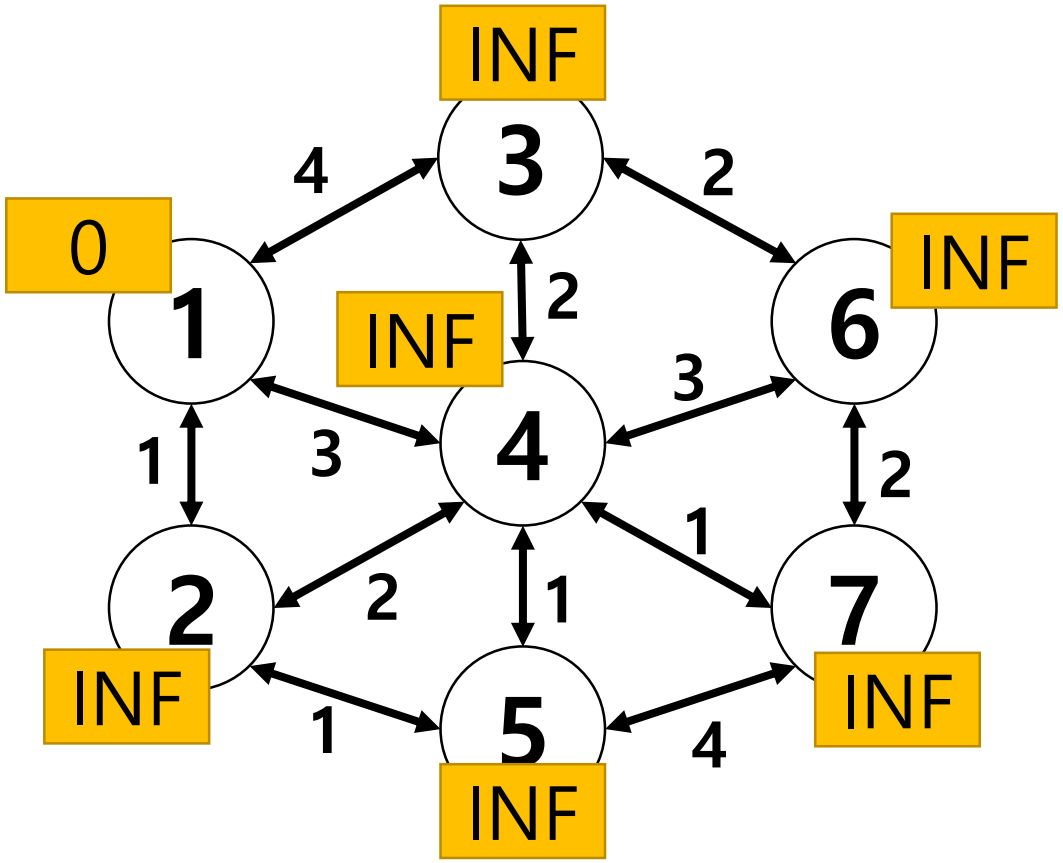
Dijkstra's Algorithm

이종서(leejseo)

최단 경로 문제

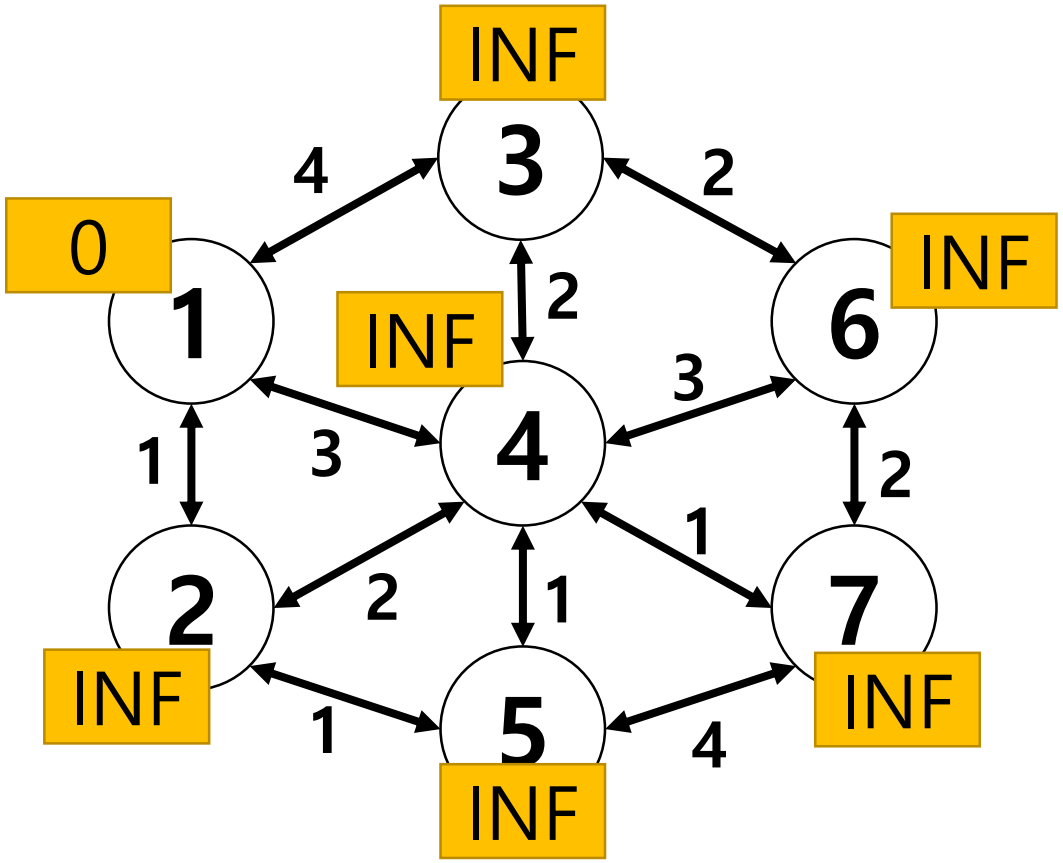
- N개의 정점과 M개의 간선으로 이루어진 그래프가 있다.
- 각 간선에 대한 정보는 시작점, 끝점, 가중치로 주어진다. (가중치 > 0)
- 시작 정점 S로 부터 다른 모든 정점까지의 최단 거리를 구해보자.

Dijkstra's Algorithm



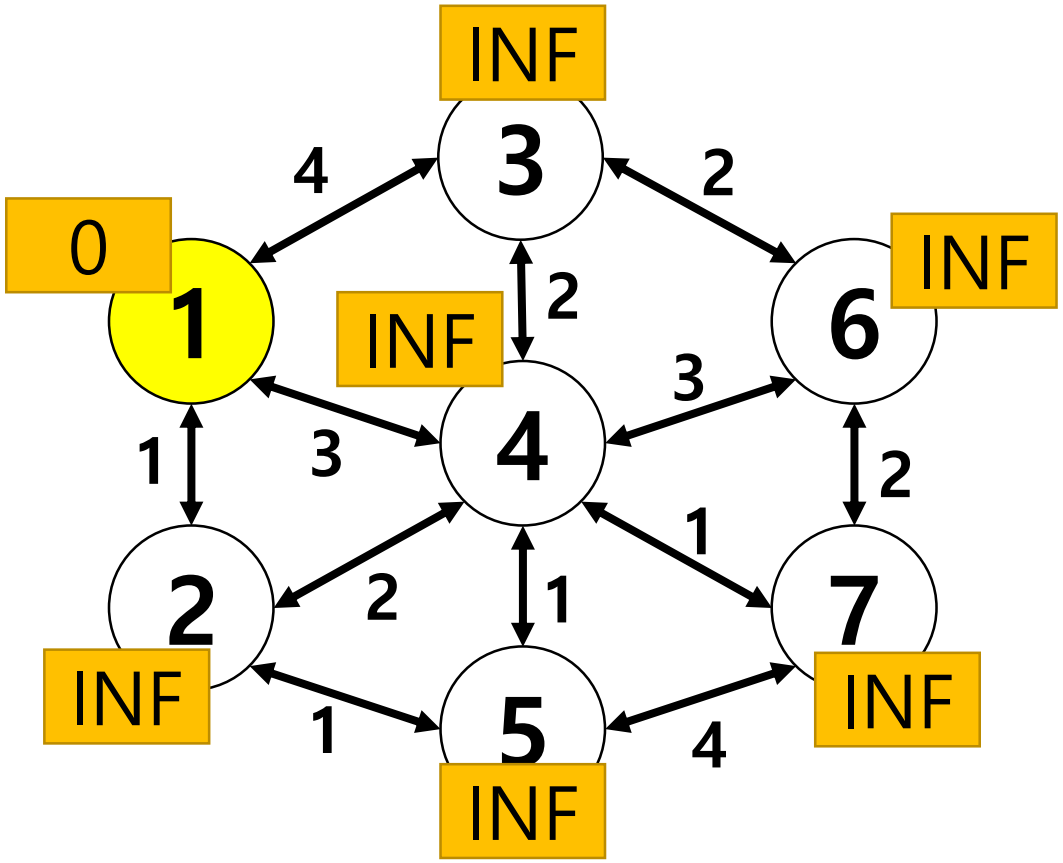
- 1번 정점으로 부터 다른 모든 정점까지의 최단 거리를 구해보자.
- 1번 정점부터 i번 정점까지의 최단 거리를 dist[i]라 하고, 이 배열을 갱신시켜 나가보자.
- 일단, dist[1] = 0이다.

Dijkstra's Algorithm



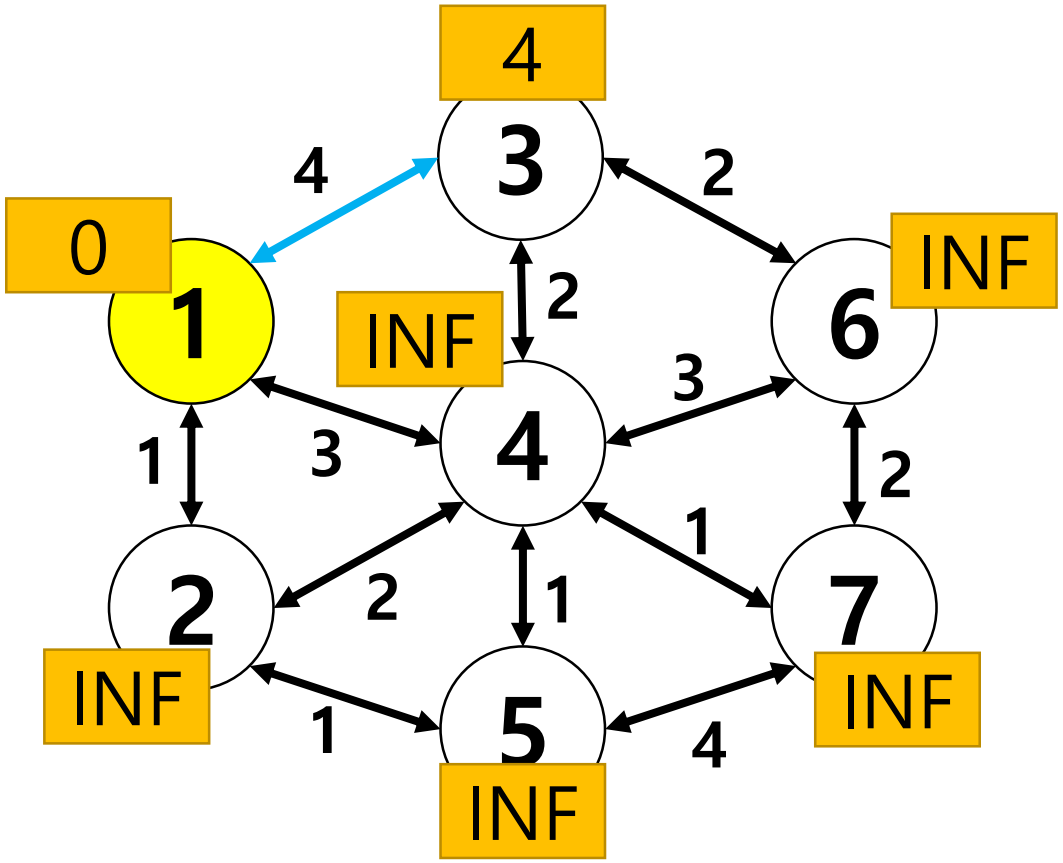
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



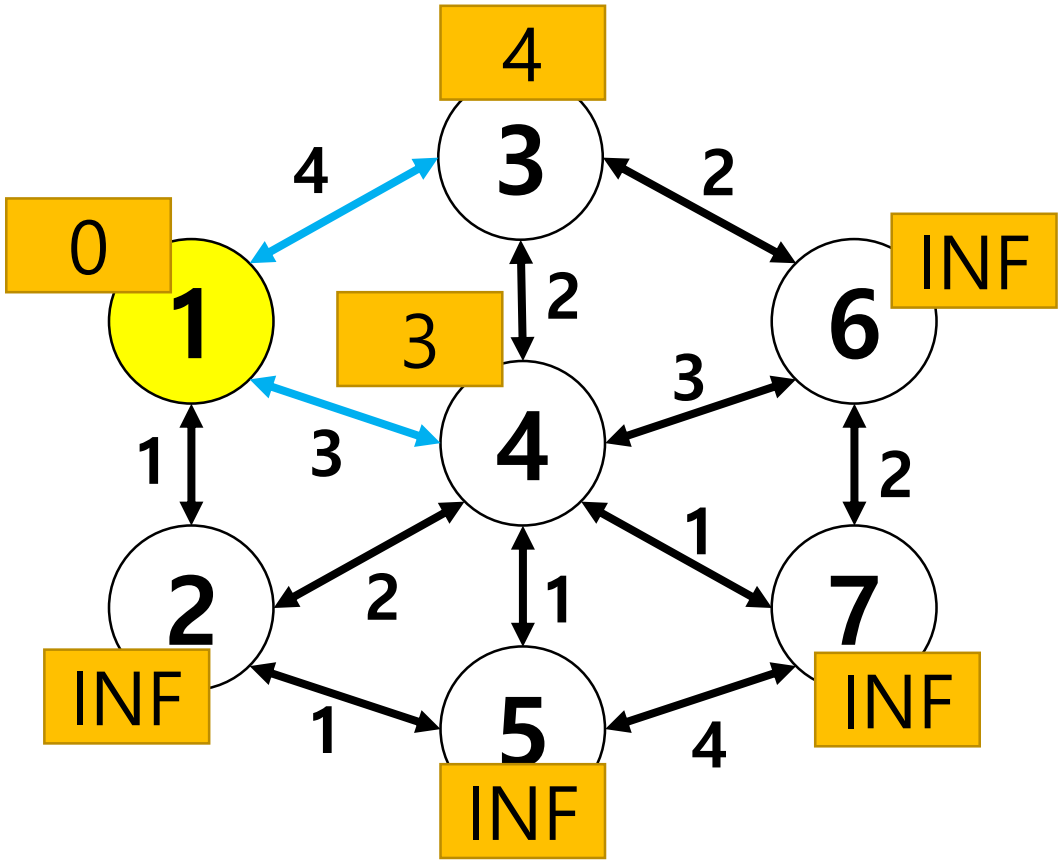
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



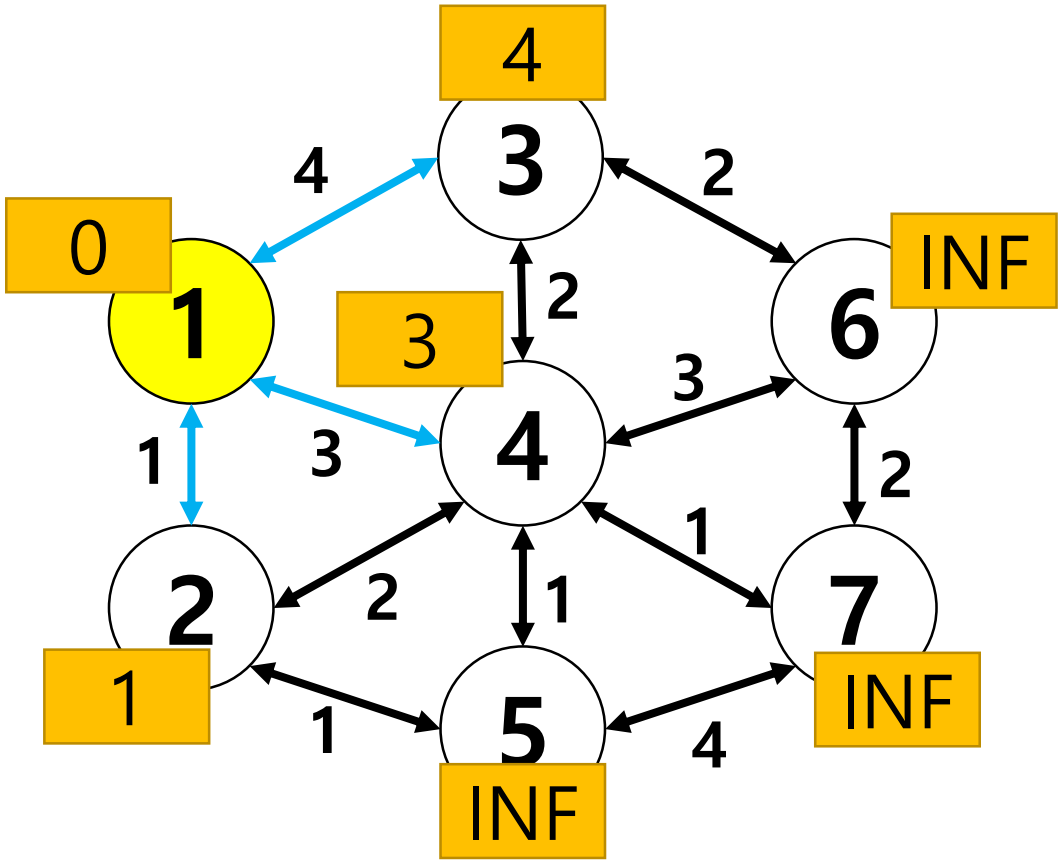
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



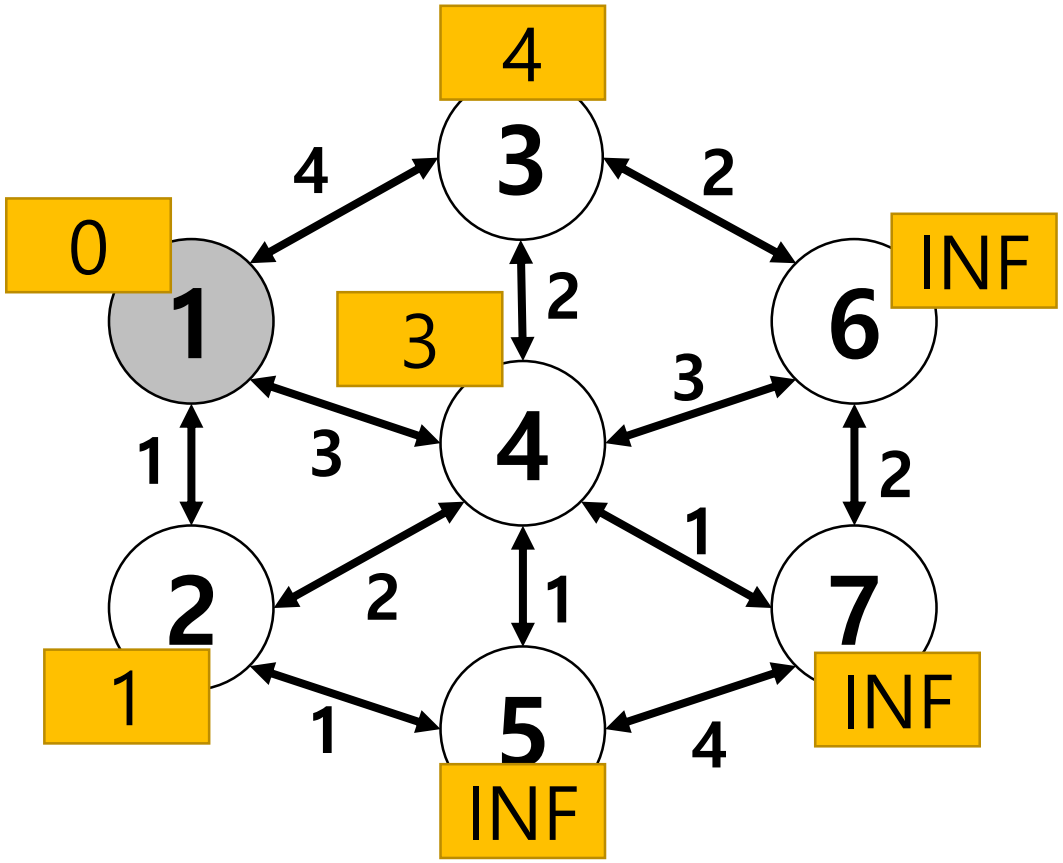
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



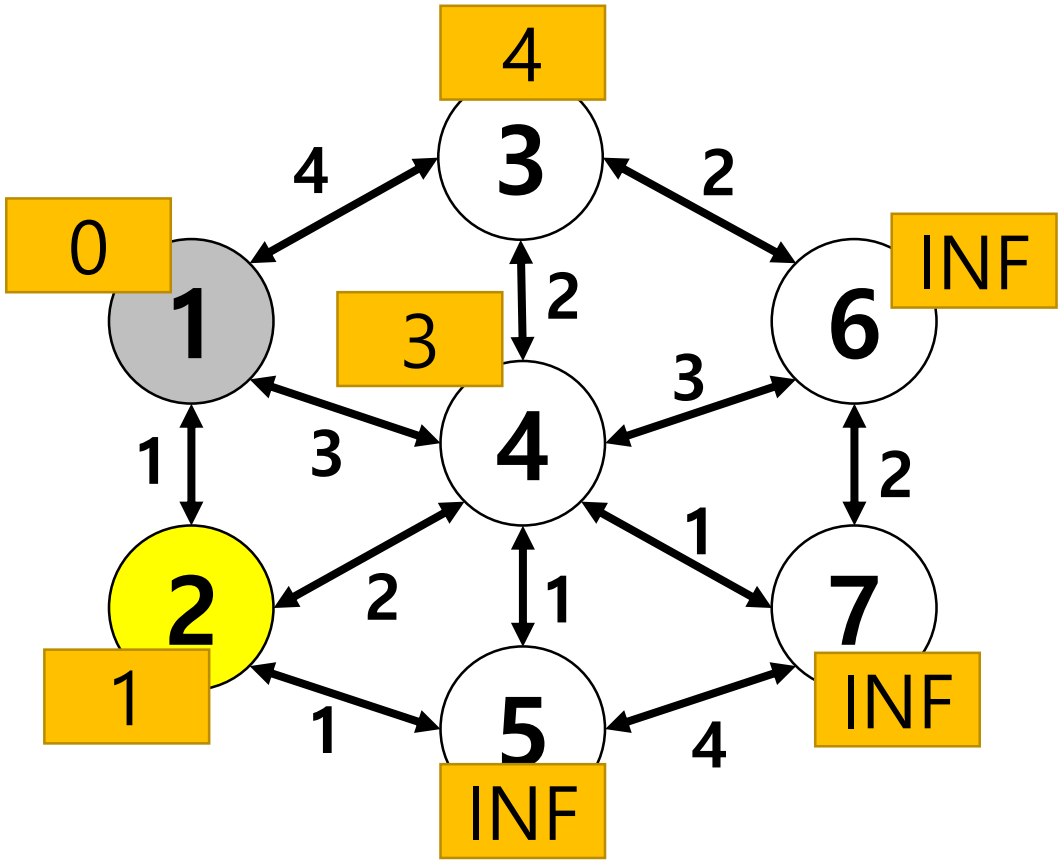
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



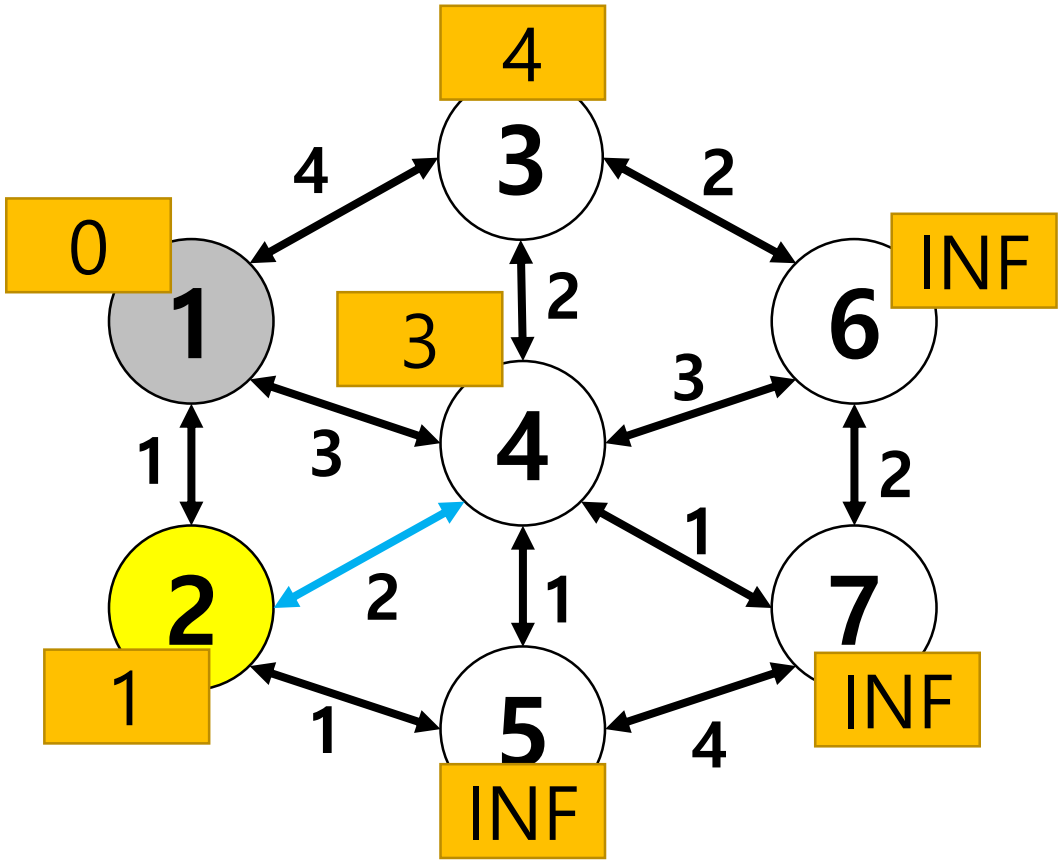
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



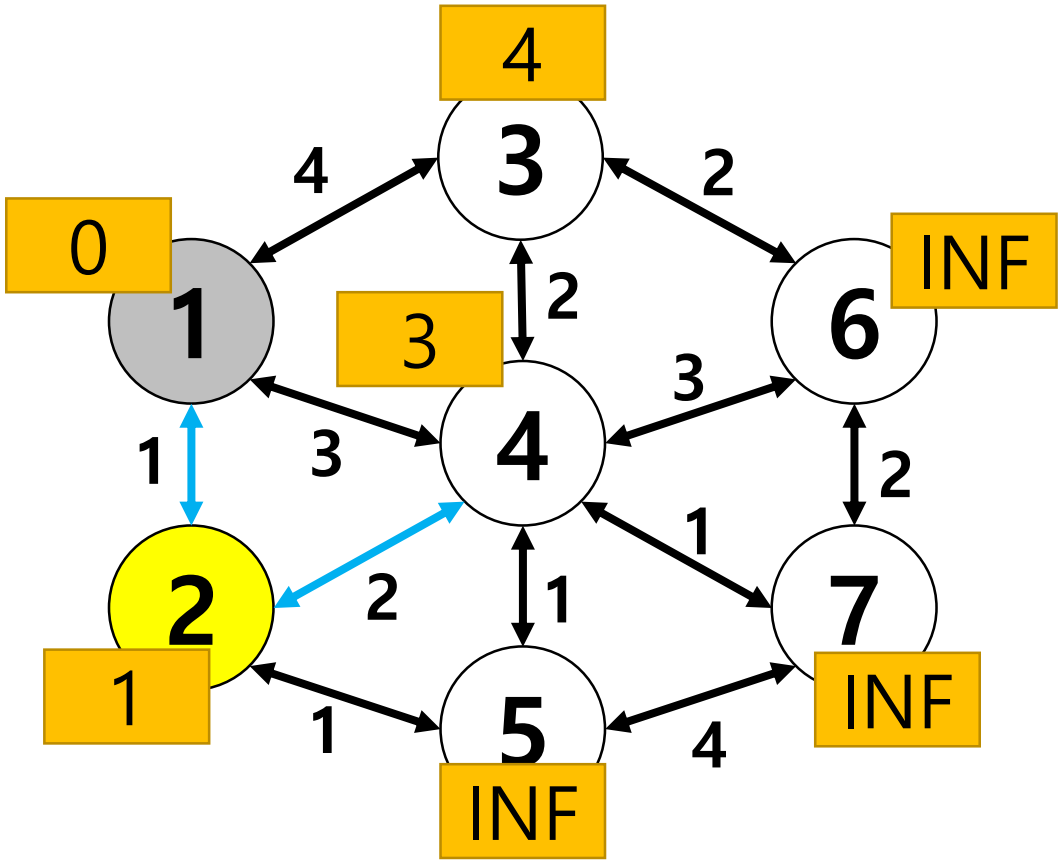
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



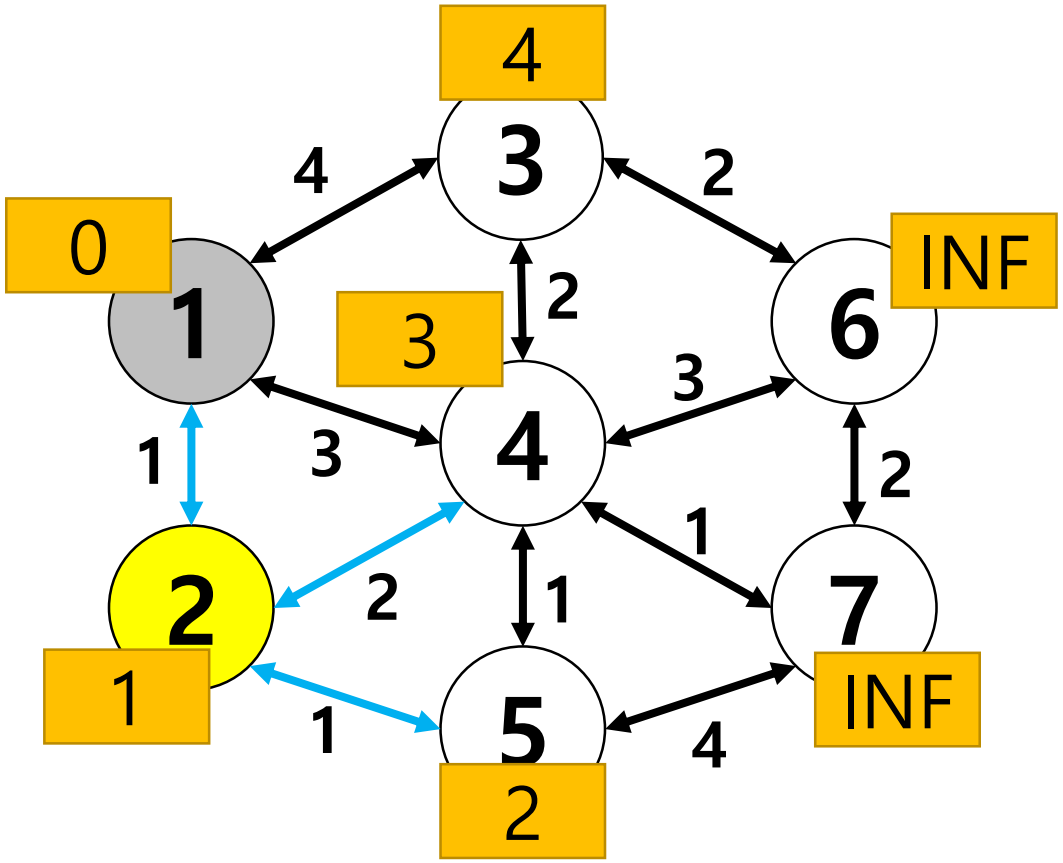
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



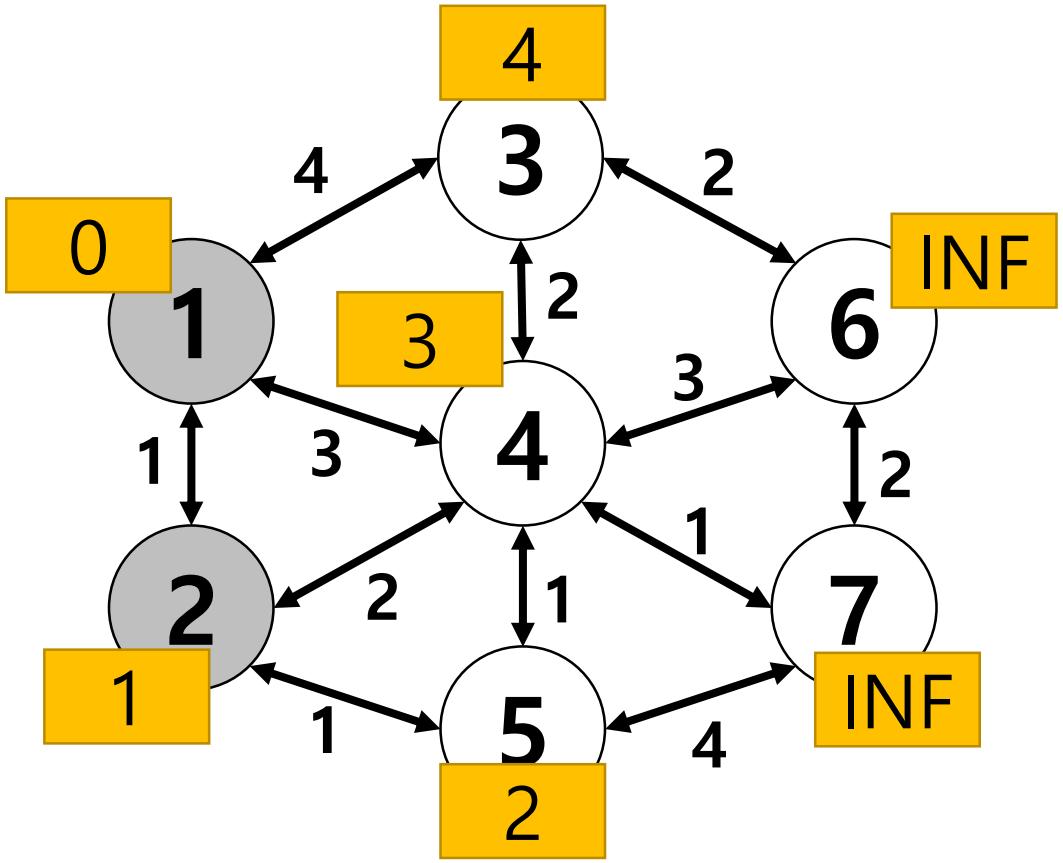
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



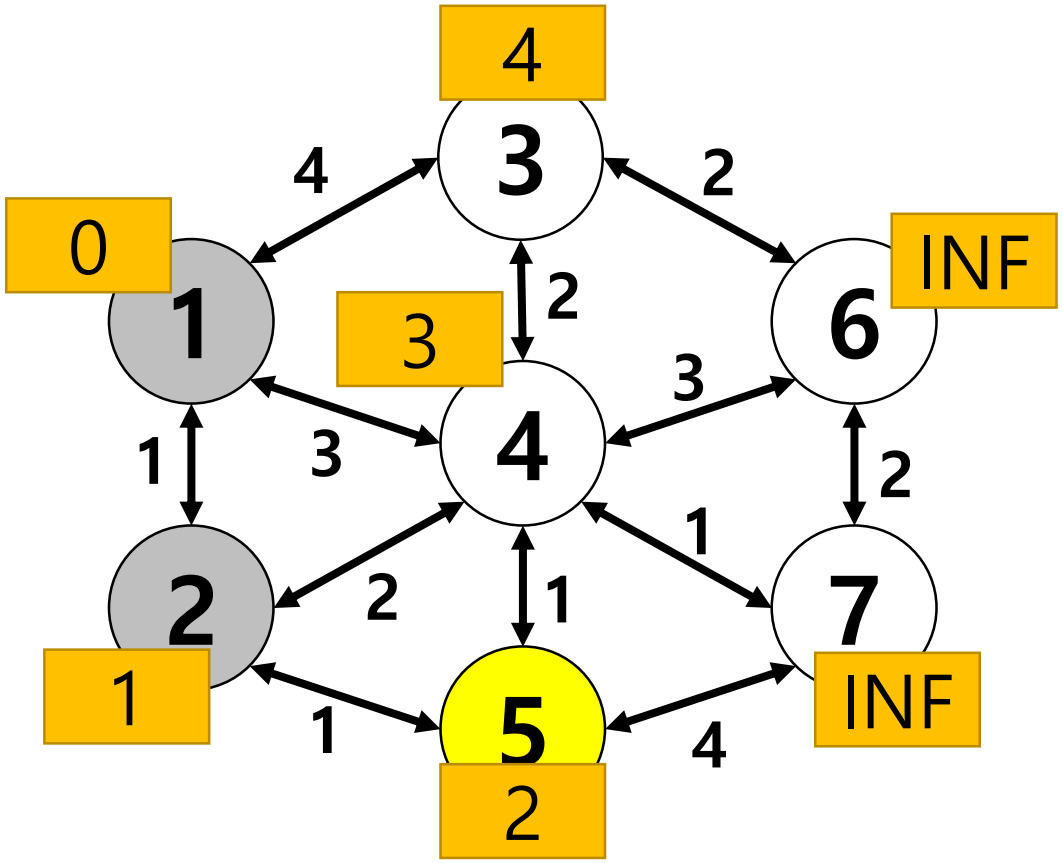
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



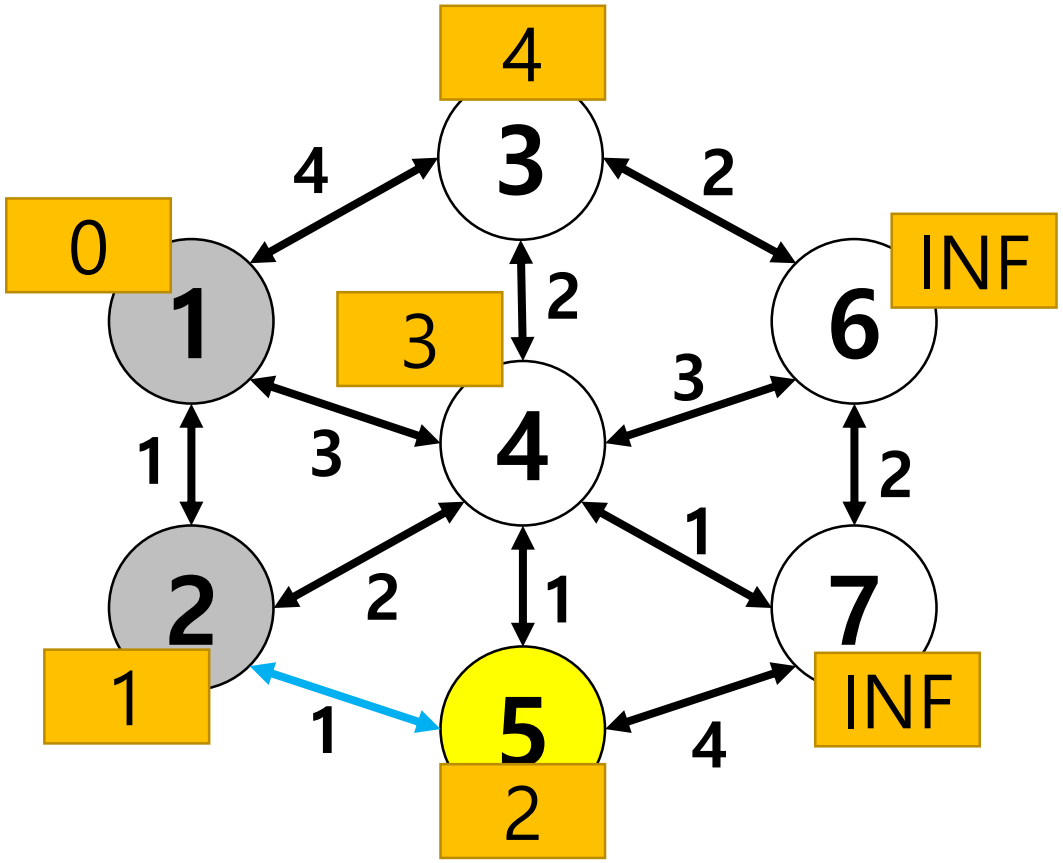
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



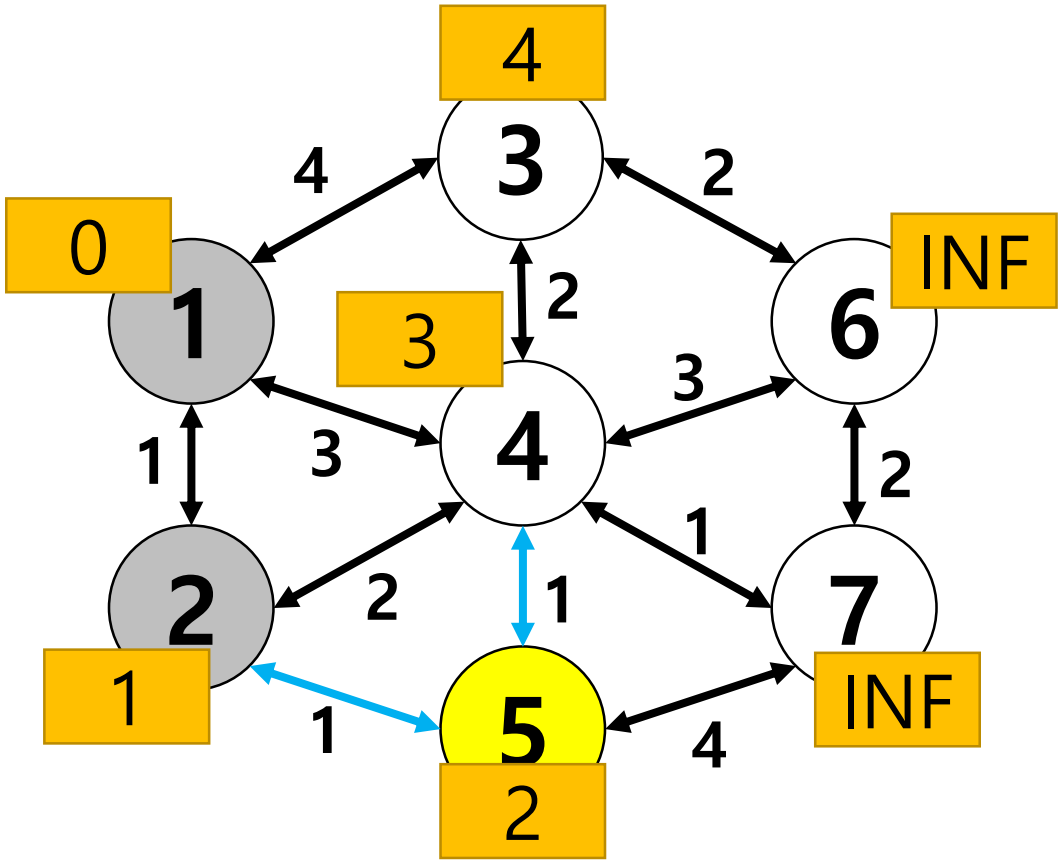
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



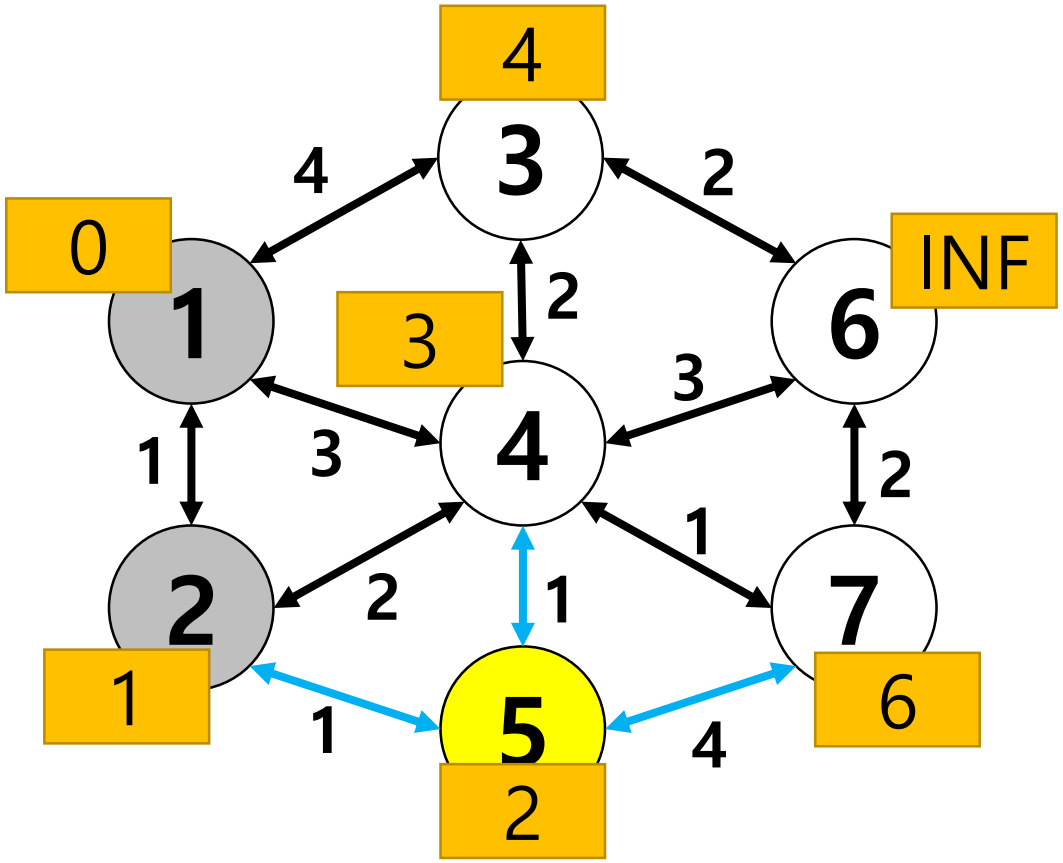
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



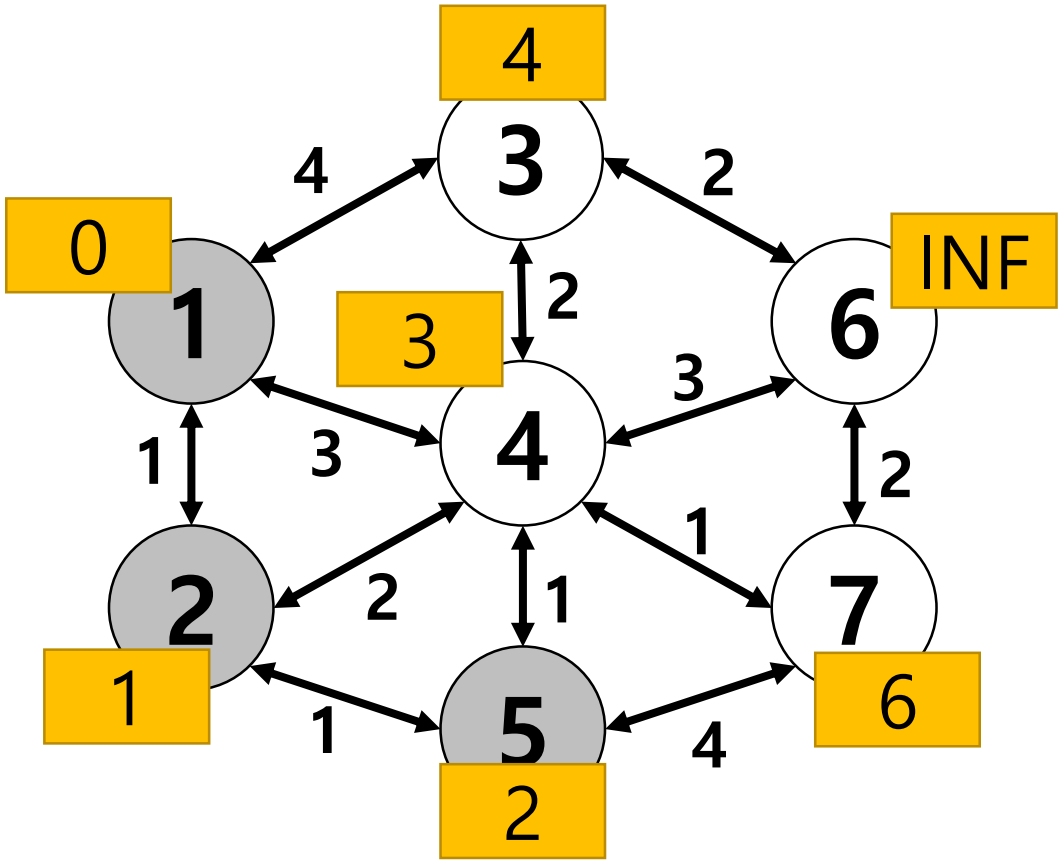
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



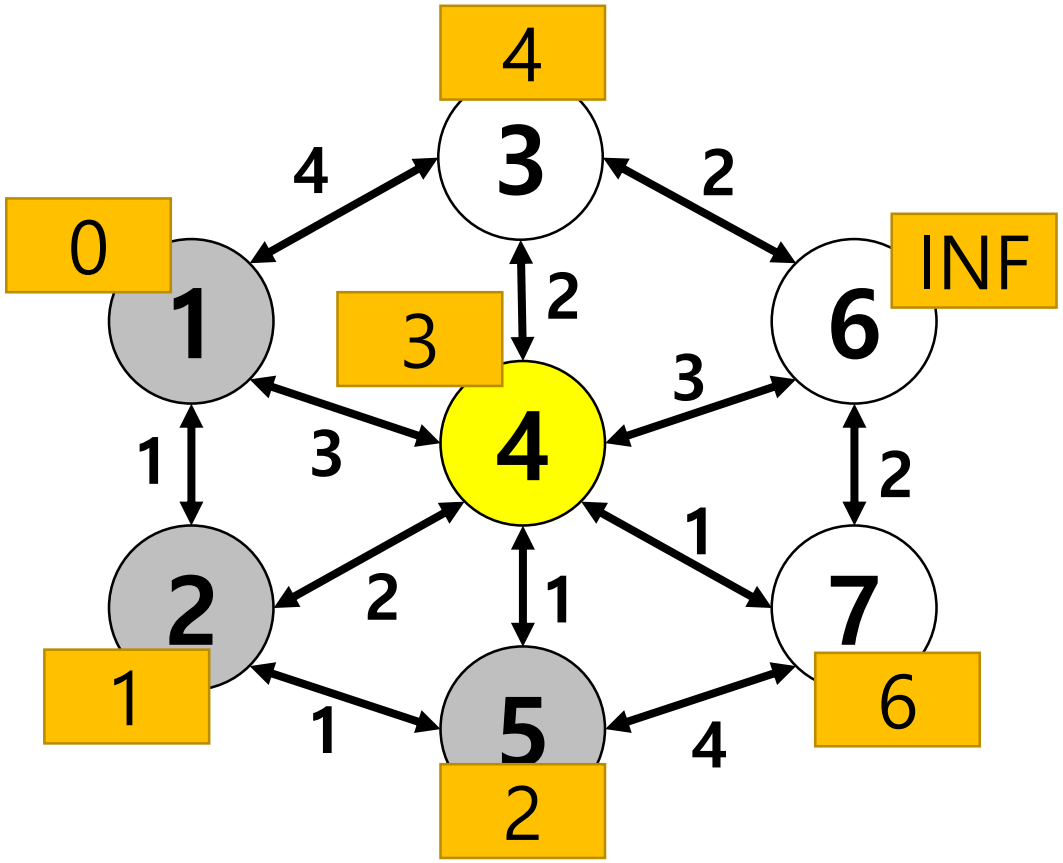
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



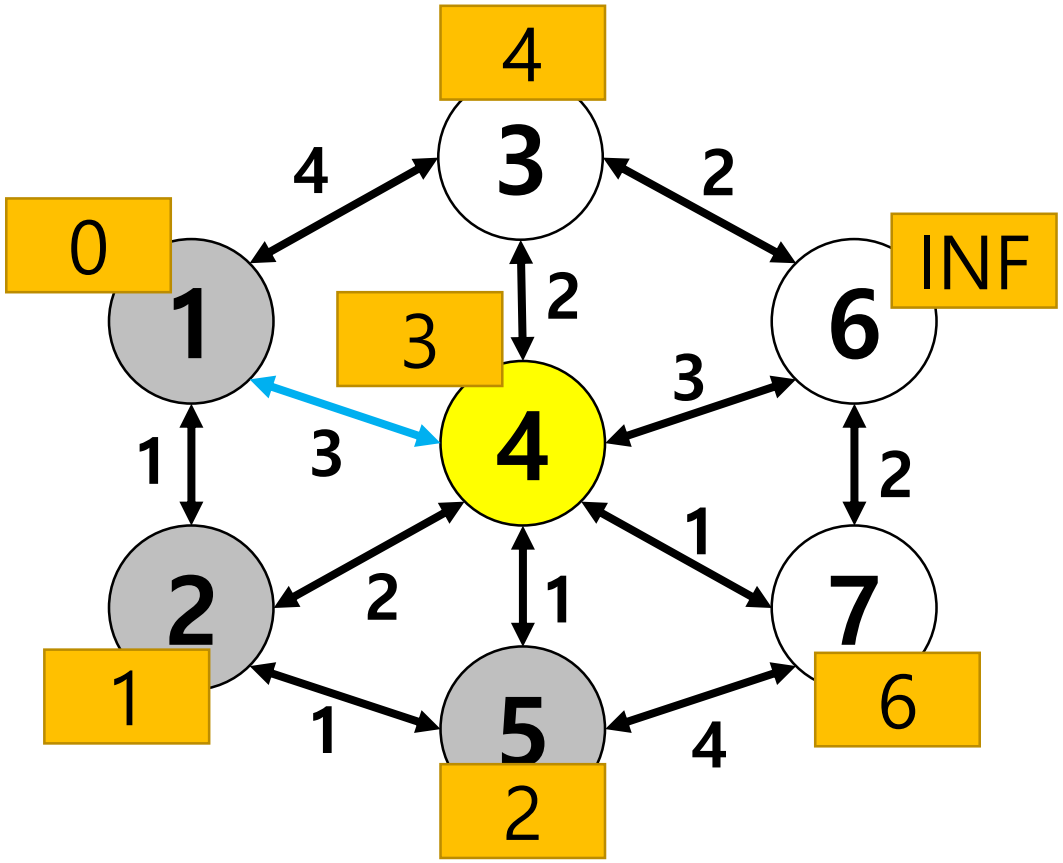
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



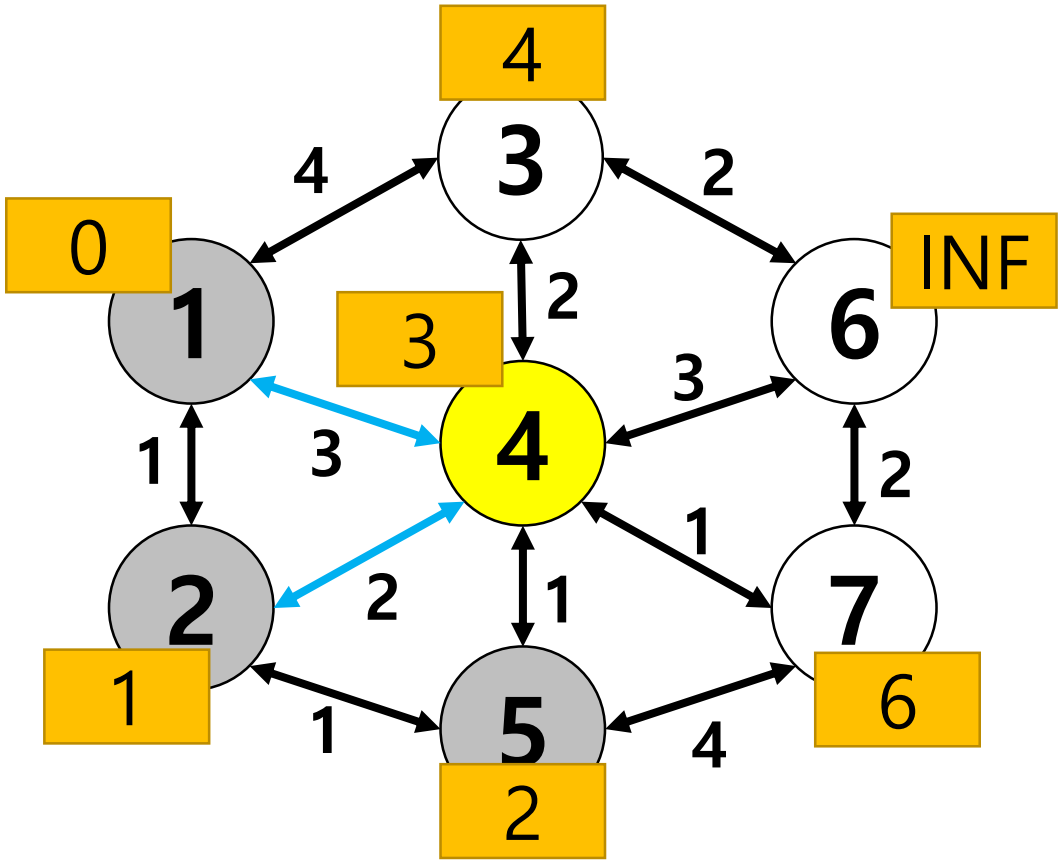
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



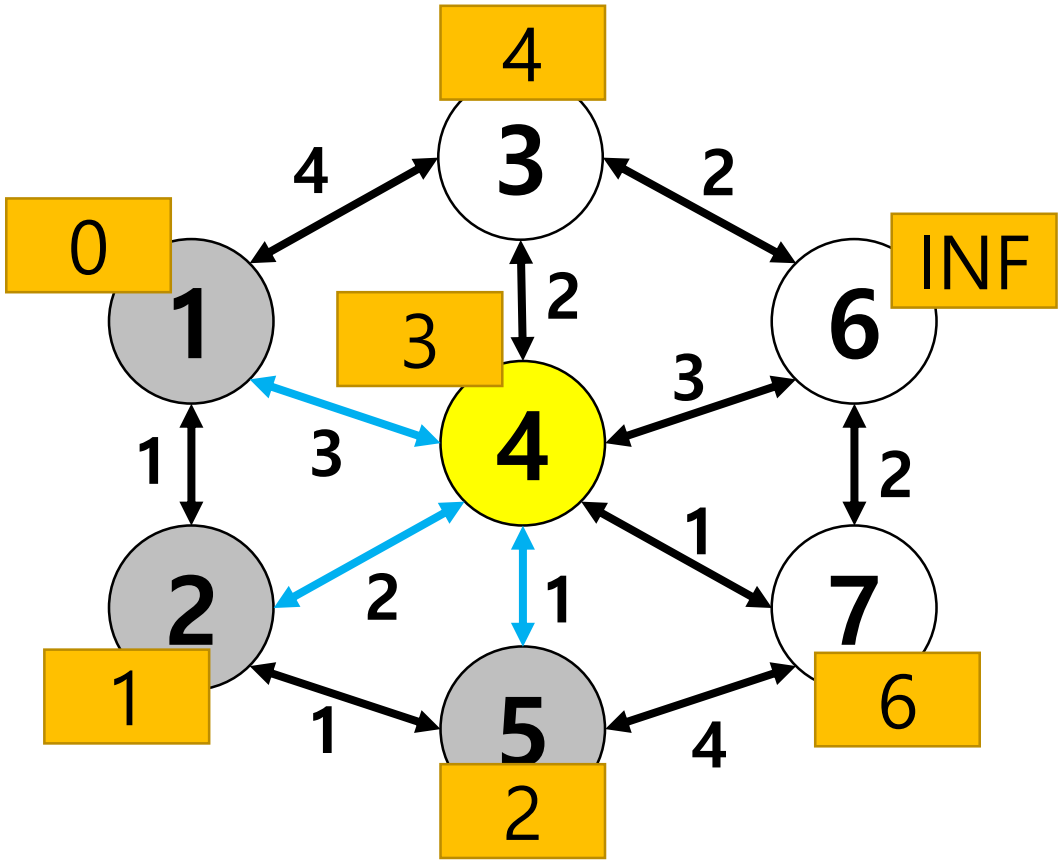
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



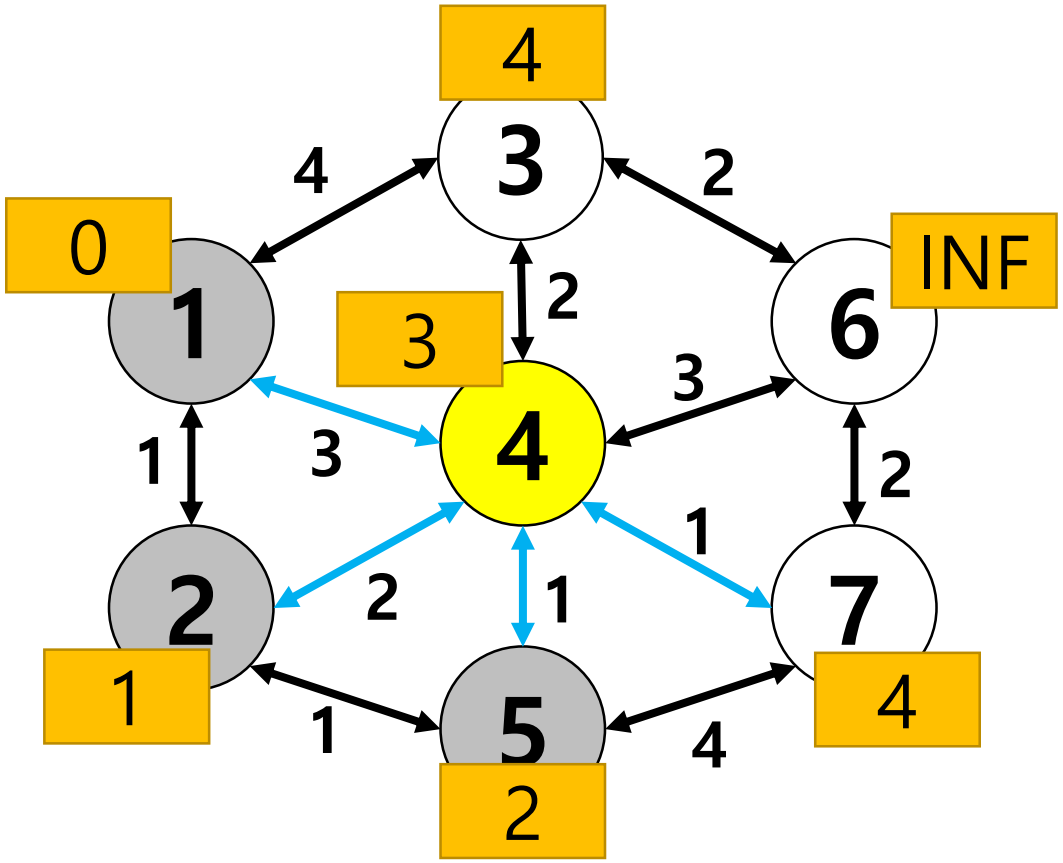
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



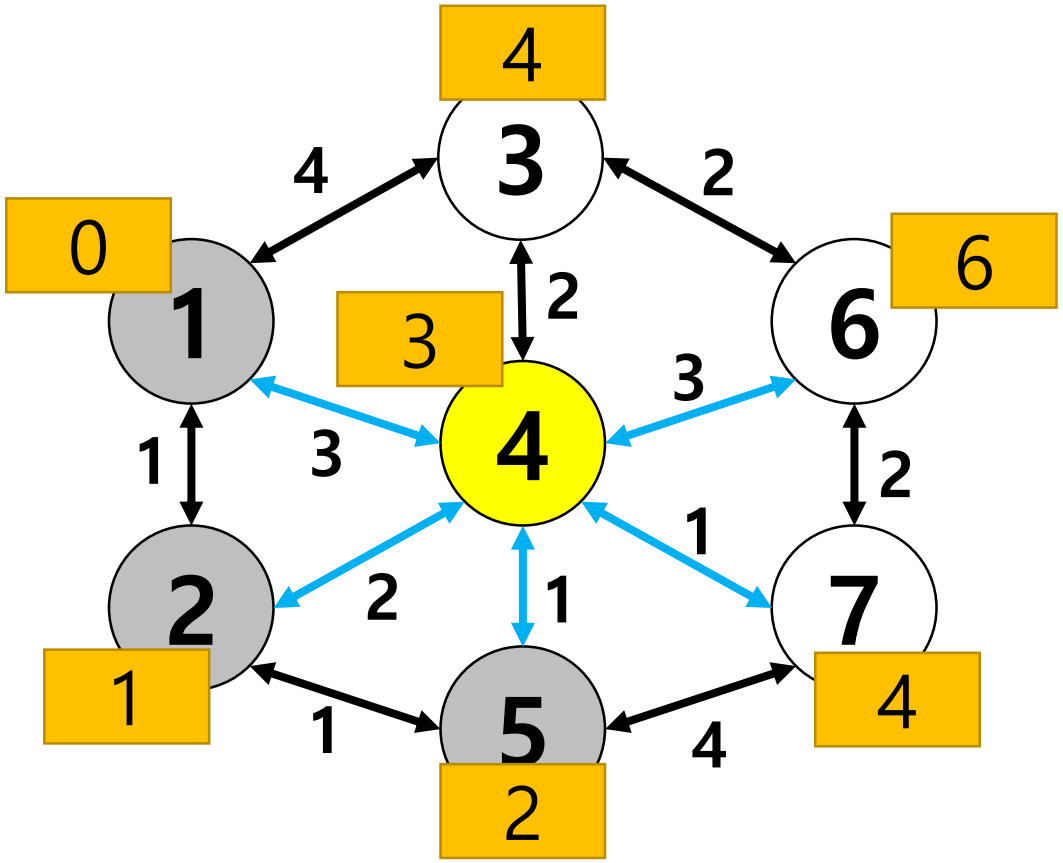
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



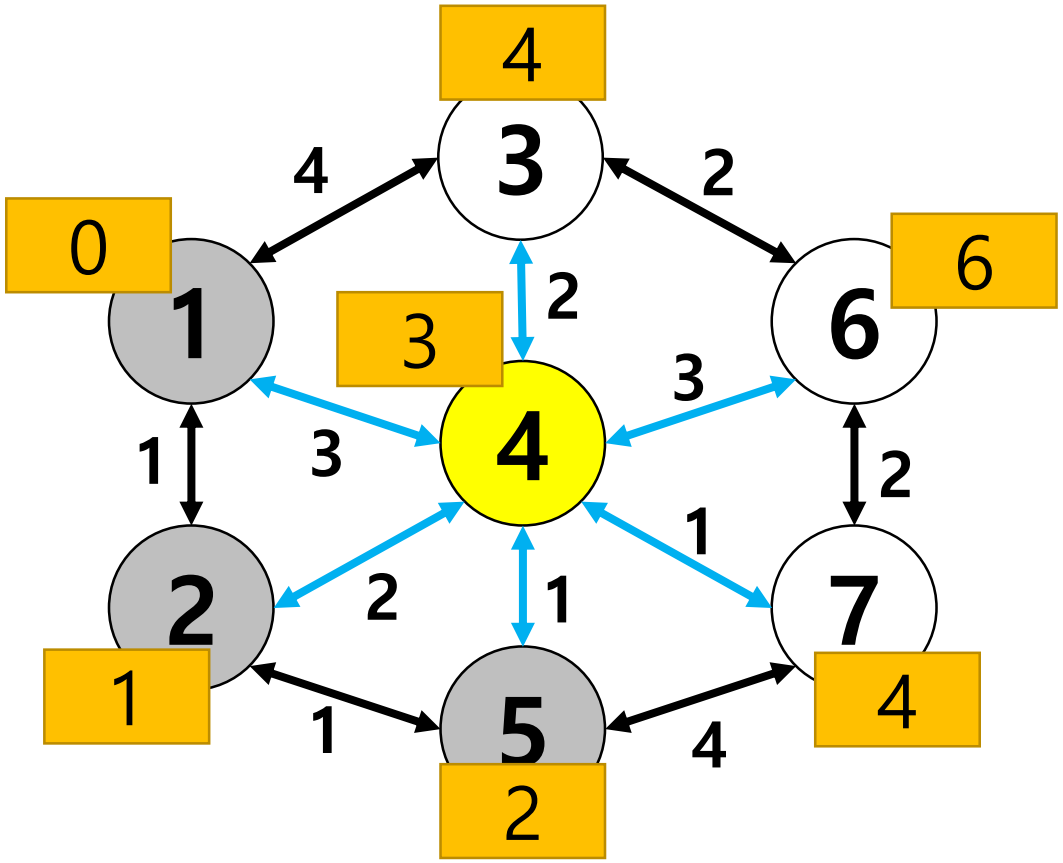
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



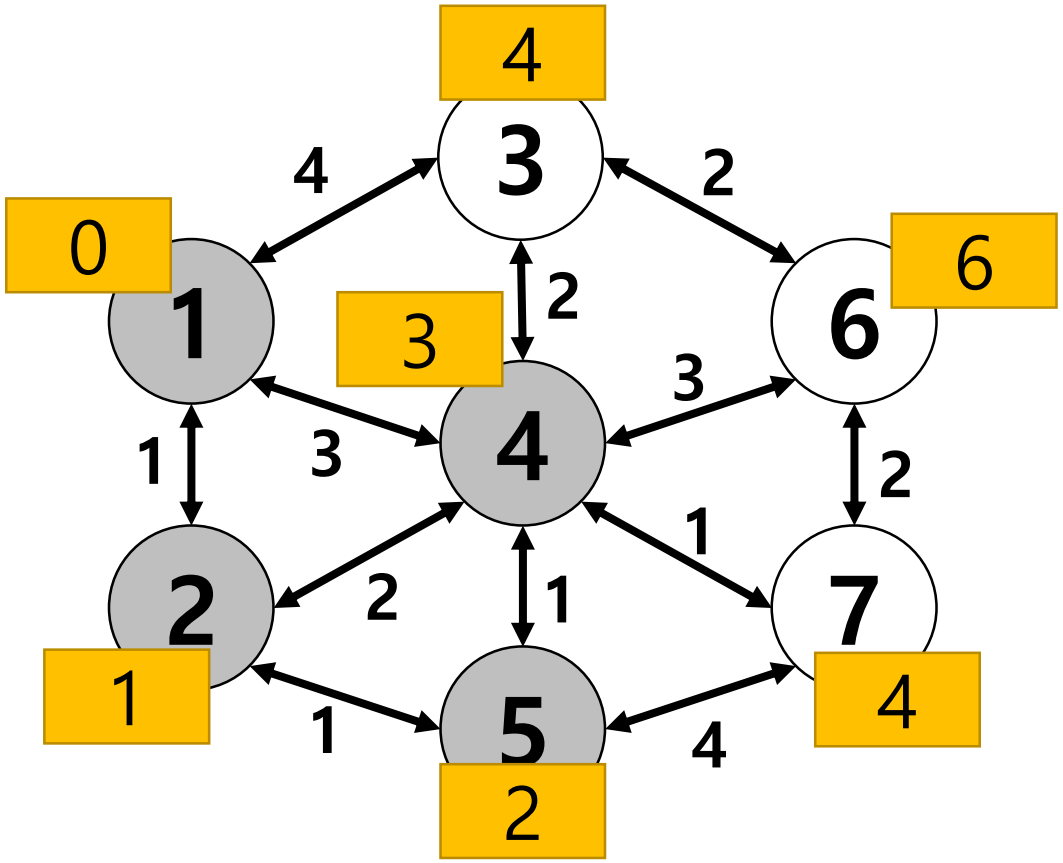
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



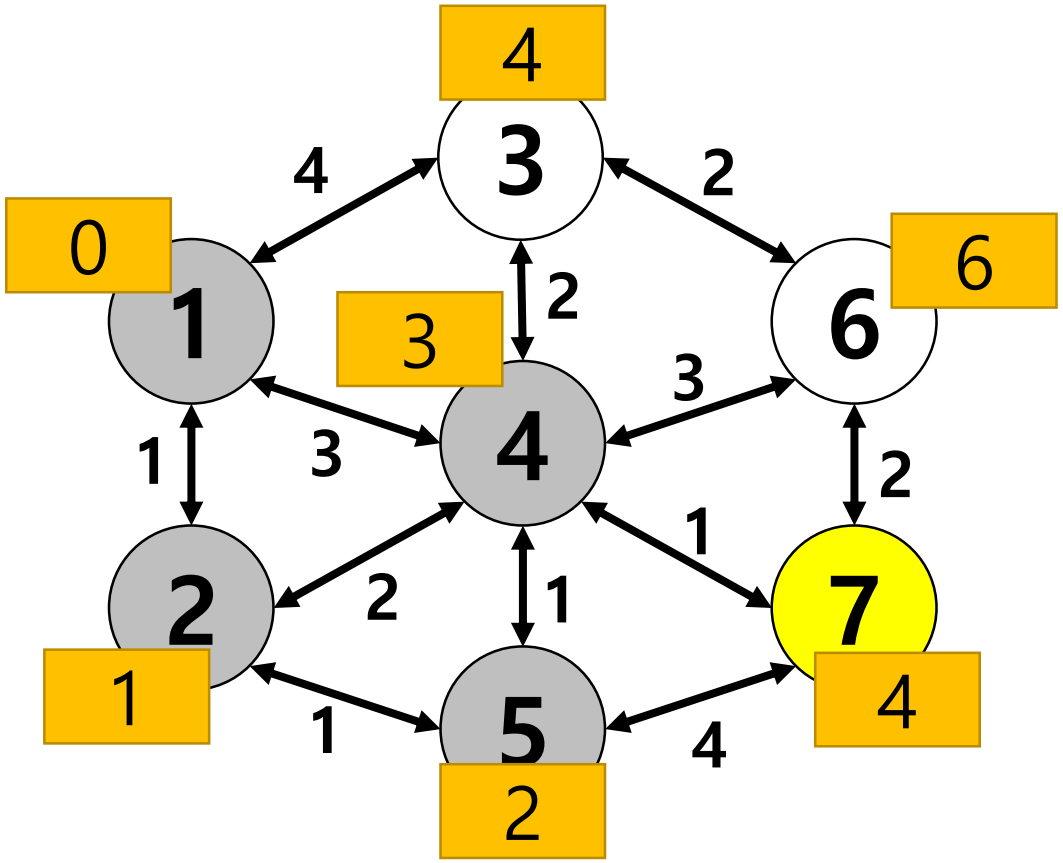
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



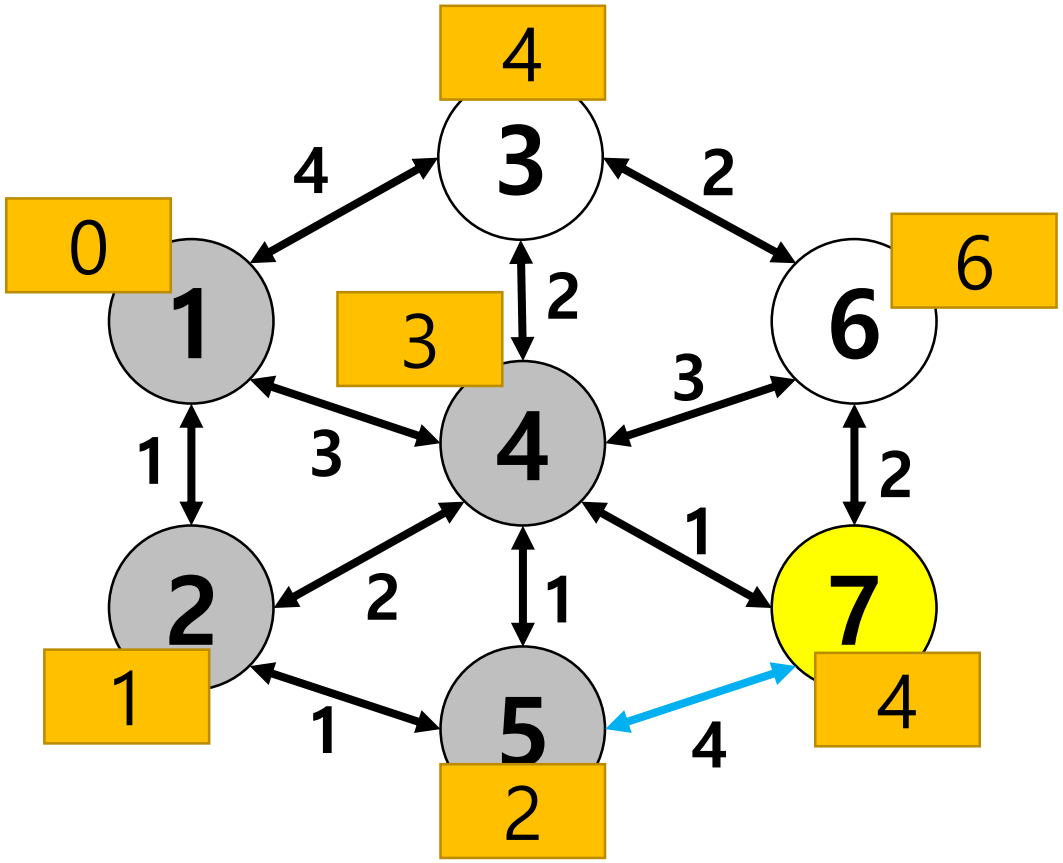
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



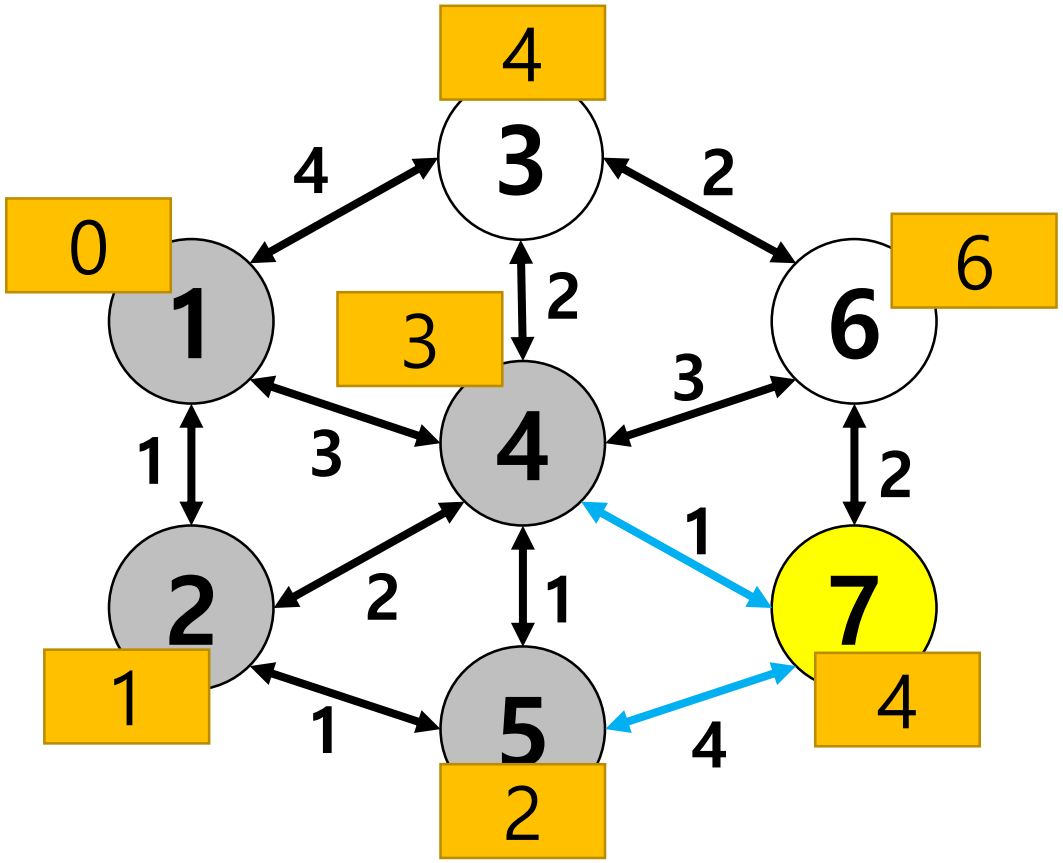
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



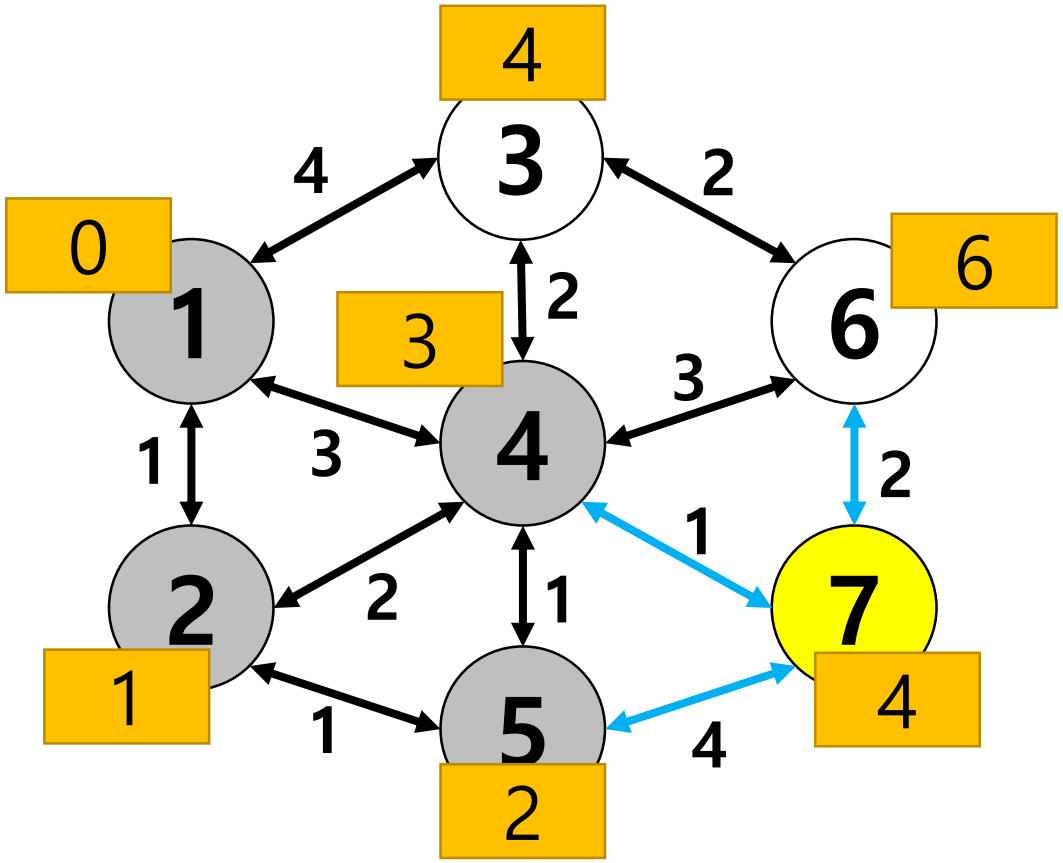
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



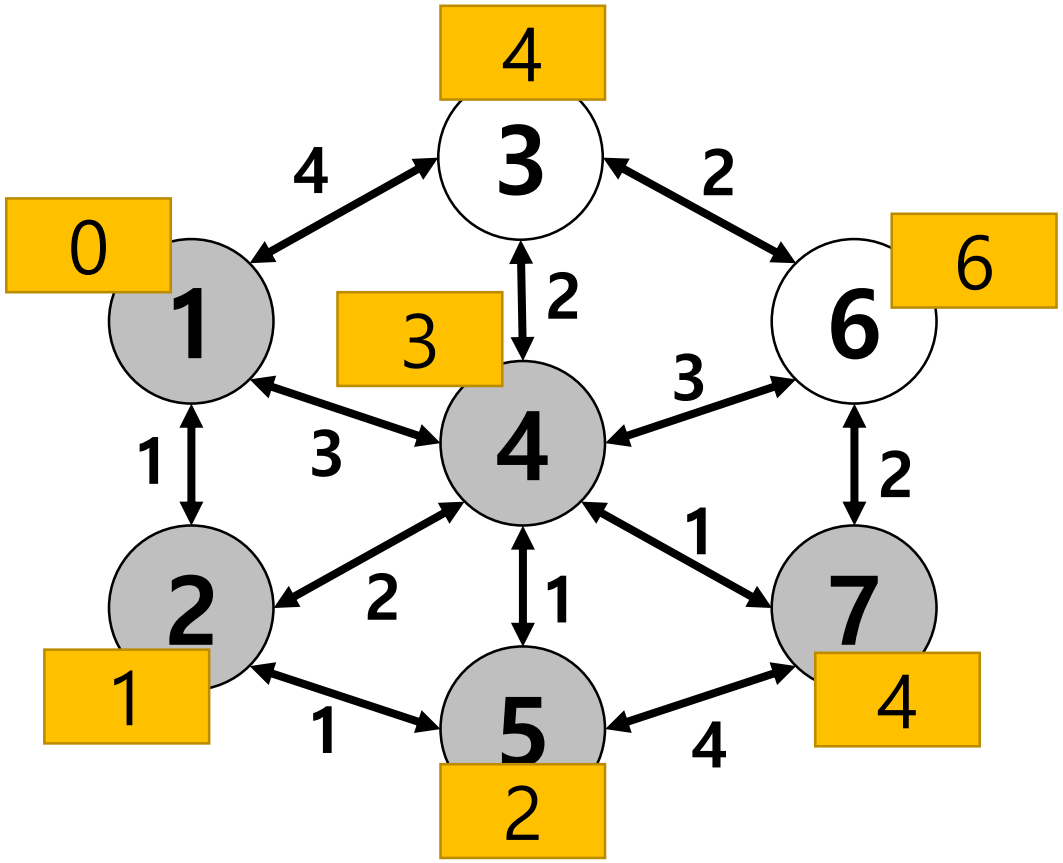
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



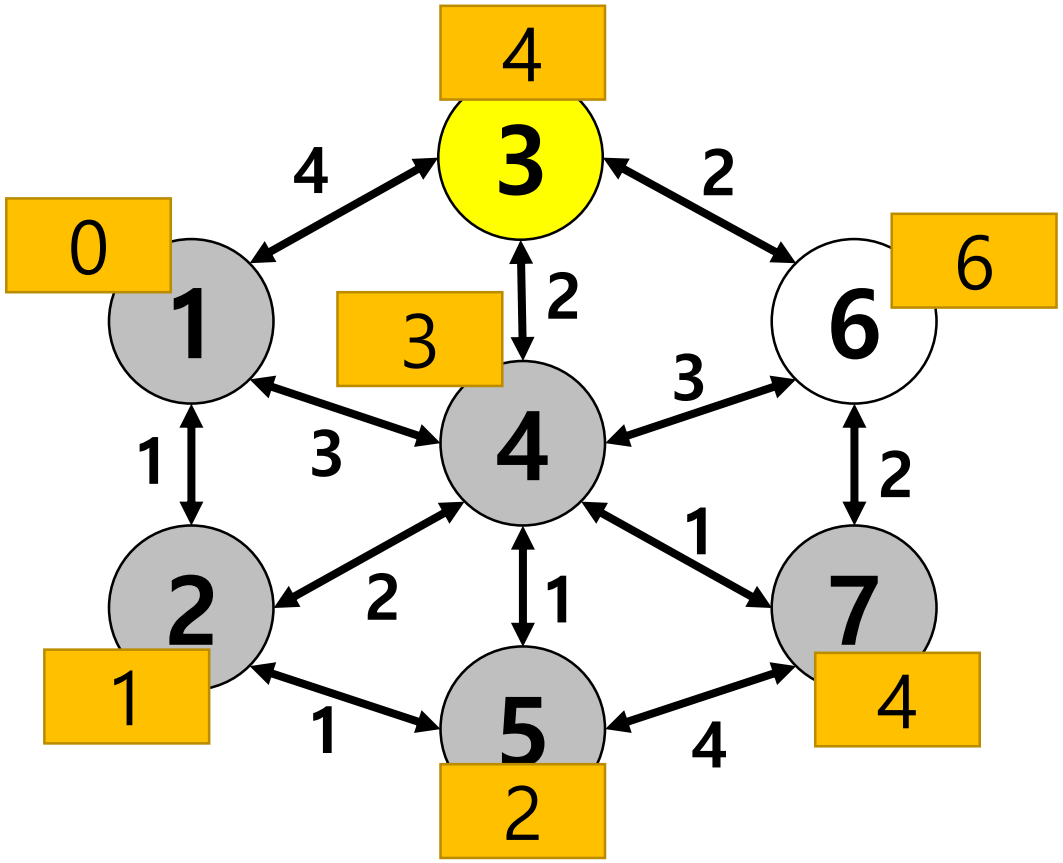
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



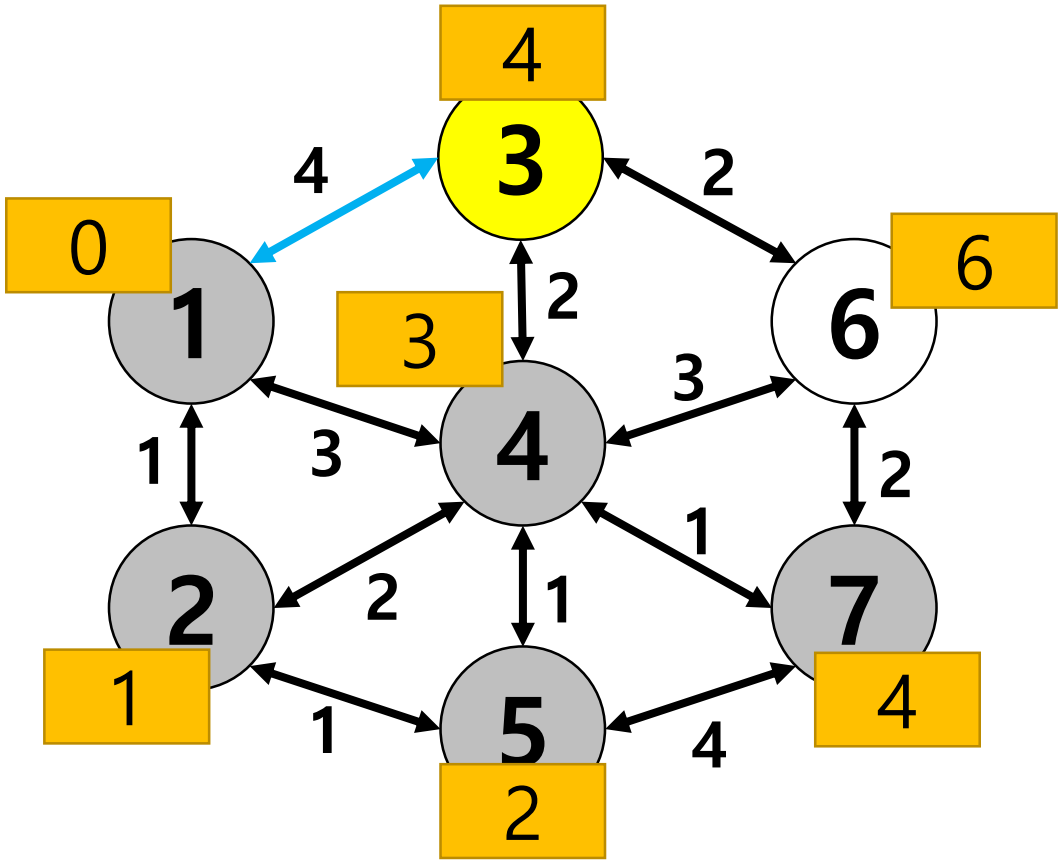
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



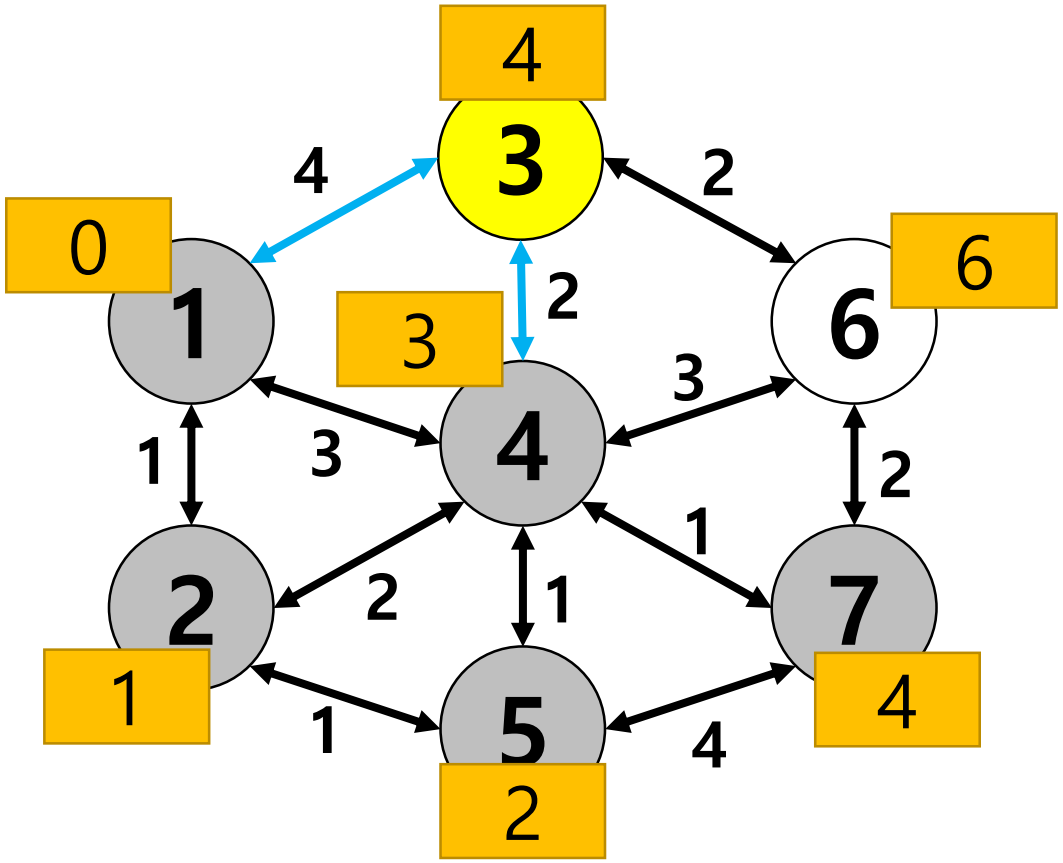
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



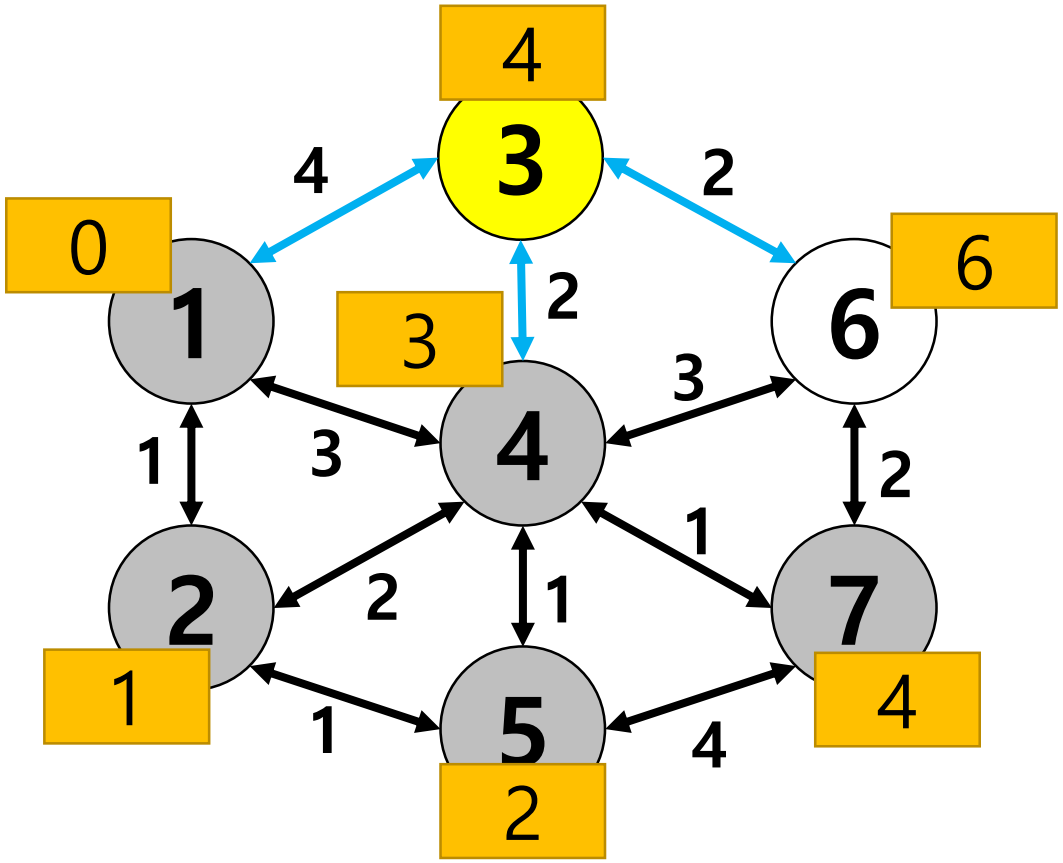
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



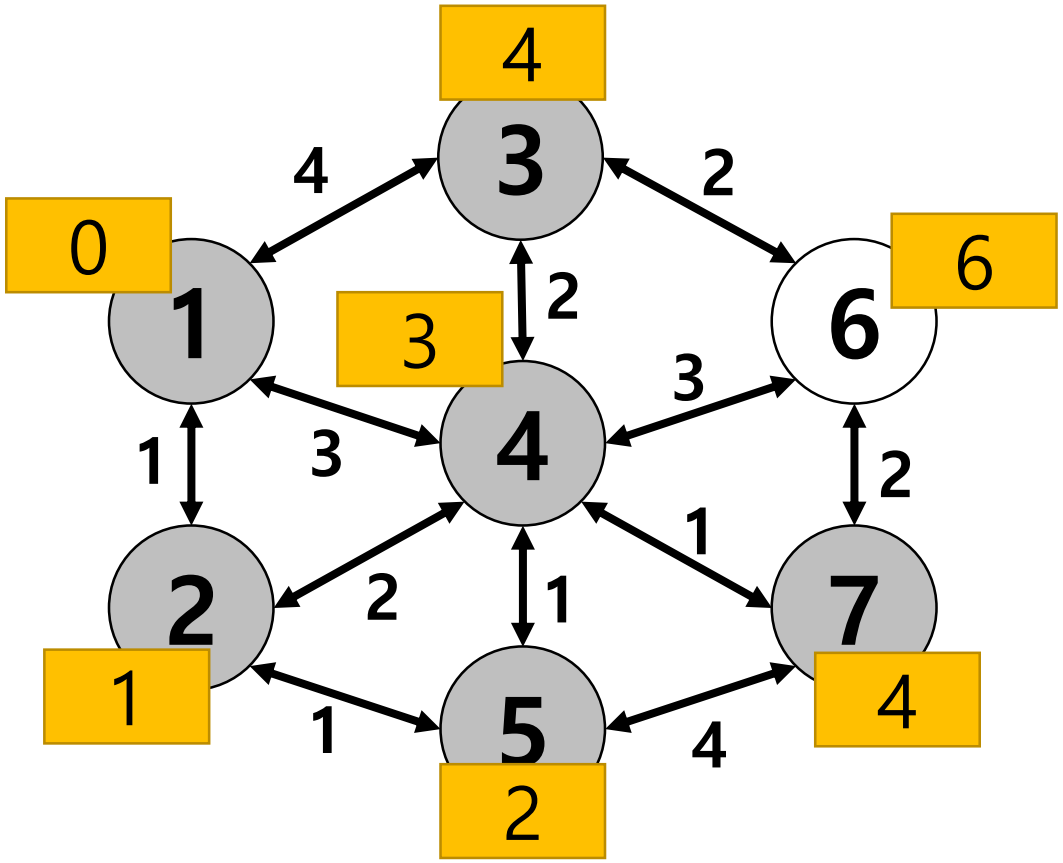
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



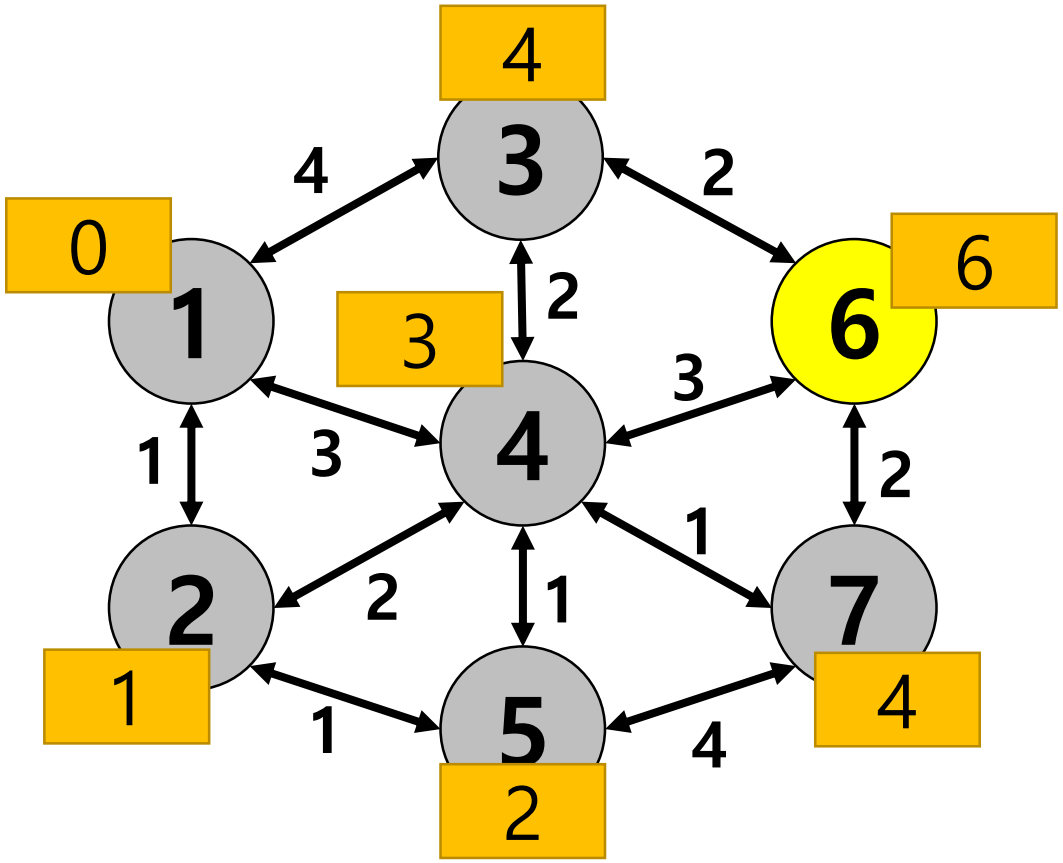
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



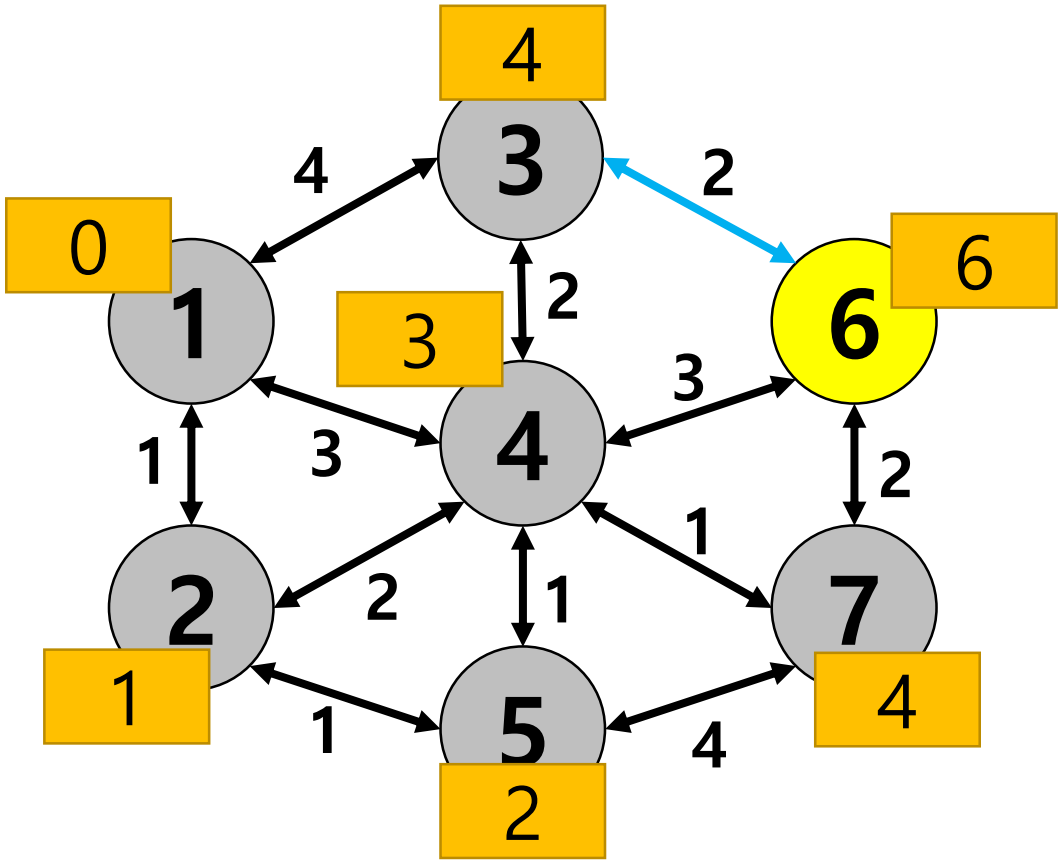
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



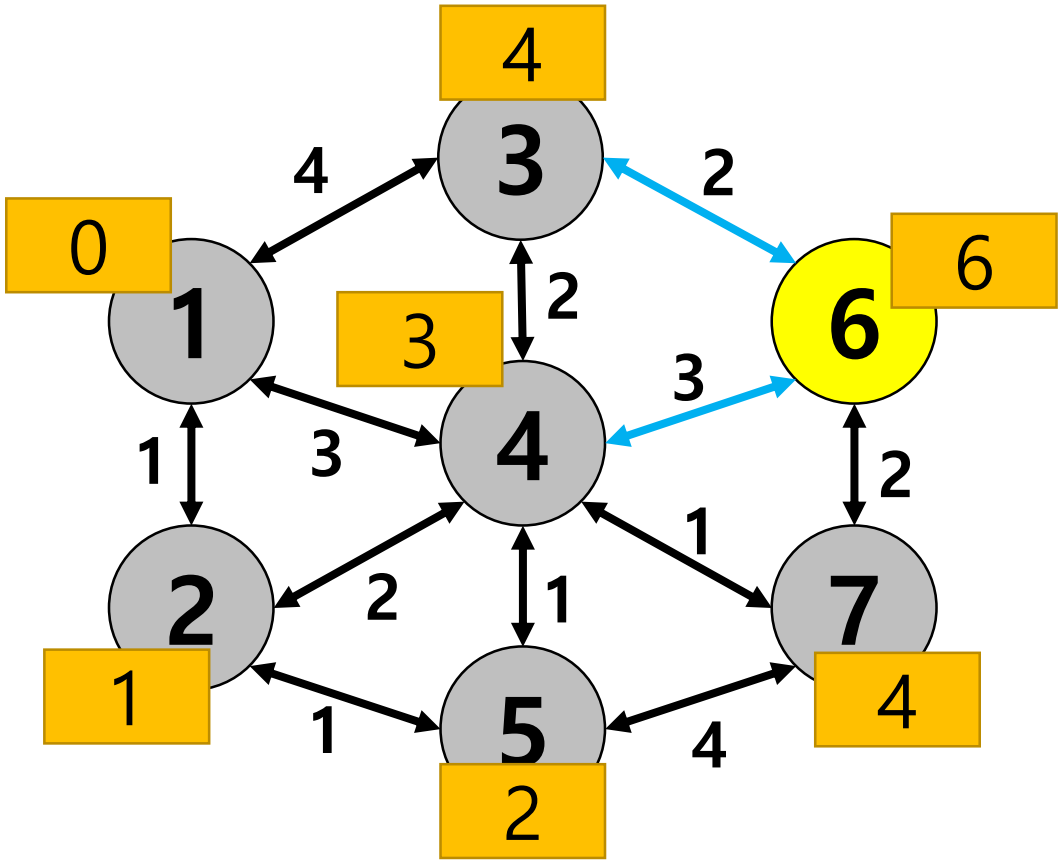
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



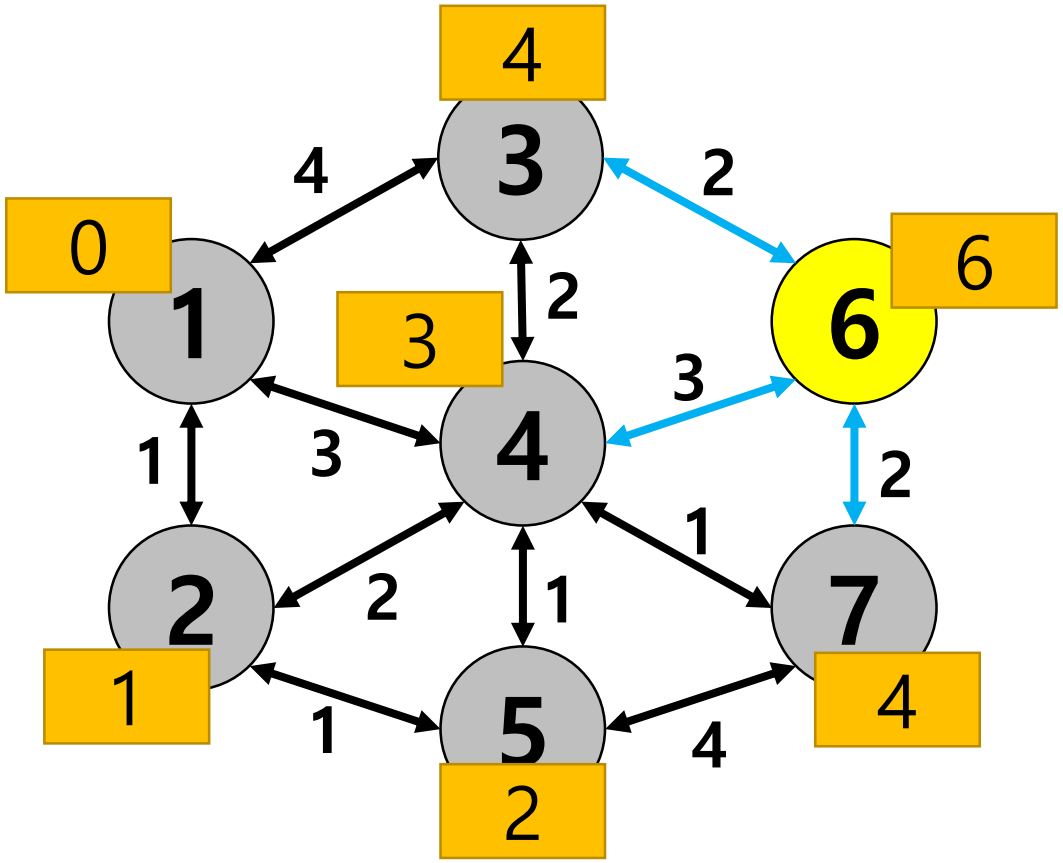
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



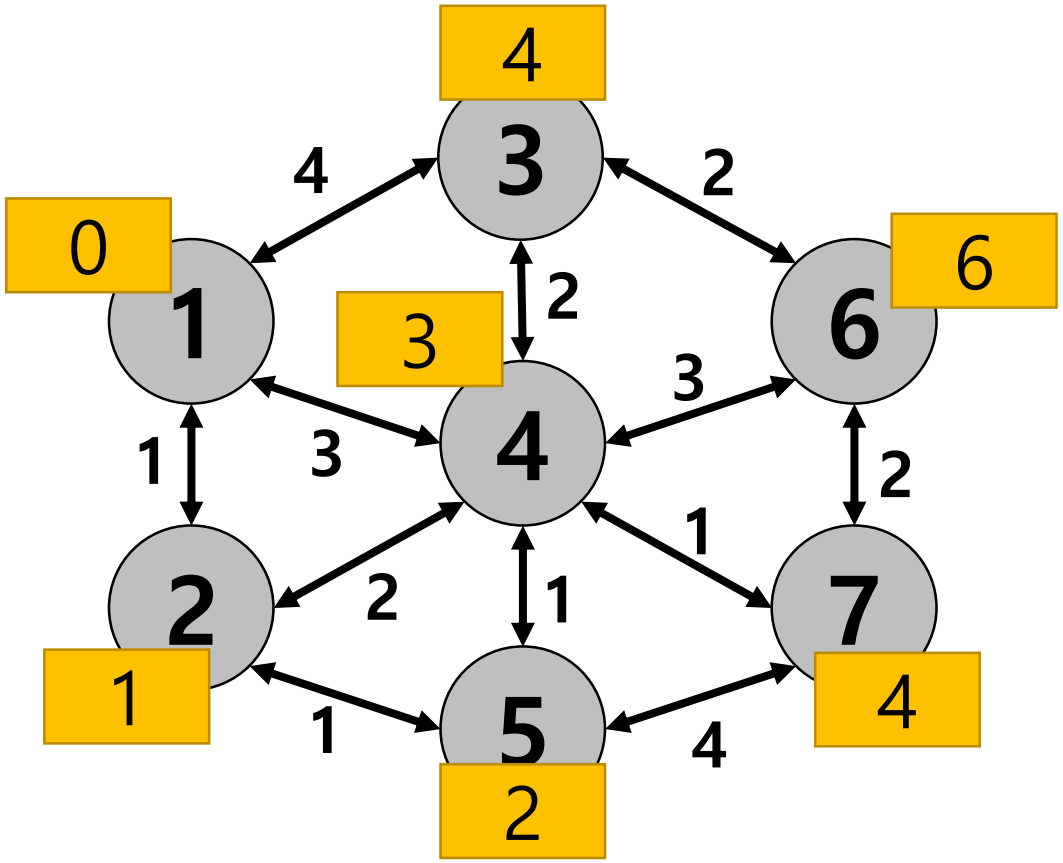
1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm



1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
3. 1-2의 과정을 반복한다.

Dijkstra's Algorithm

1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
   (최솟값 찾기 O(N))
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
   (인접한 간선들을 훑음 O(deg(i)))
3. 1-2의 과정을 반복한다.
   (1+2를 1번 수행하는데 O(N), 이를 N번 수행하므로 O(N^2))

Dijkstra's Algorithm

1. 최단거리가 확정되지 않은 노드 중 dist[i] 값이 가장 작은 노드를 고른다.
   (최솟값 찾기 O(N) → Heap을 쓰면 O(log N))
2. 해당 노드의 dist[i] 값을 확정시키고, 인접한 노드들의 dist[i] 값을 갱신한다.
   (인접한 간선들을 훑음 O(deg(i)))
3. 1-2의 과정을 반복한다.
   (1+2를 1번 수행하는데 O(N), 이를 N번 수행하므로 O(N^2)
   → 1을 M번 수행하므로 O(M log N), 2는 총 합해서 O(M)
   → Heap을 이용하면 Dijkstra's Algorithm은 O(M log N))

BOJ 11779 – 최소비용 구하기 2

- 요약: 최단 거리를 구한 후, 역추적을 통해 실제 최단 경로 또한 구하는 문제

BOJ 11779 – 최소비용 구하기 2

- 요약: 최단 거리를 구한 후, 역추적을 통해 실제 최단 경로 또한 구하는 문제

- 끝 정점으로 부터 시작하여 “직전 정점”을 계속 찾아나가자.

BOJ 11779 – 최소비용 구하기 2

- 요약: 최단 거리를 구한 후, 역추적을 통해 실제 최단 경로 또한 구하는 문제

- 끝 정점으로 부터 시작하여 "직전 정점"을 계속 찾아나가자.
- y의 직전 정점 (중 하나가 될 수 있는) x 찾는 법:
  → y에 인접한 정점들을 보며 dist[y] = dist[x] + weight(x, y)인 x 찾기

BOJ 2307 – 도로검문

- 요약: 간선을 하나 끊어 최단 거리를 최대화 하는 문제

BOJ 2307 – 도로검문

- 요약: 간선을 하나 끊어 최단 거리를 최대화 하는 문제
- 모든 간선을 끊어본다면, O(M)번 다익스트라 알고리즘을 돌리므로, O(M^2 log N) → 시간 초과

BOJ 2307 – 도로검문

- 요약: 간선을 하나 끊어 최단 거리를 최대화 하는 문제
- 모든 간선을 끊어본다면, O(M)번 다익스트라 알고리즘을 돌리므로, O(M^2 log N) → 시간 초과

- **핵심 관찰:** 최단 경로 밖의 간선을 끊더라도, 최단 경로는 여전히 최단 경로이다.

BOJ 2307 – 도로검문

- 요약: 간선을 하나 끊어 최단 거리를 최대화 하는 문제
- 모든 간선을 끊어본다면, O(M)번 다익스트라 알고리즘을 돌리므로, O(M^2 log N) → 시간 초과

- **핵심 관찰:** 최단 경로 밖의 간선을 끊더라도, 최단 경로는 여전히 최단 경로이다.
- → 최단 경로에 속하는 간선만 끊어보면 됨. O(NM log N)

BOJ 20313 – 출퇴근

- **핵심 관찰:** 윤이가 마법을 써서 가중치들을 바꾸고 나면, 다시 되돌릴 수 없다.

BOJ 20313 – 출퇴근

- **핵심 관찰:** 윤이가 마법을 써서 가중치들을 바꾸고 나면, 다시 되돌릴 수 없다.
- "다시 되돌릴 수 없음"을 어떻게 그래프로 표현할까?

BOJ 20313 – 출퇴근

- **핵심 관찰:** 윤이가 마법을 써서 가중치들을 바꾸고 나면, 다시 되돌릴 수 없다.
- "다시 되돌릴 수 없음"을 어떻게 그래프로 표현할까?
- 각 간선이 $T_{*,k}$ 의 가중치를 가지는 그래프를 G_k라 할 때, 각 G_k를 하나의 레이어로 하여 그래프를 만들고, 각 레이어 사이를 가중치 0의 directed edge로 이어준다!