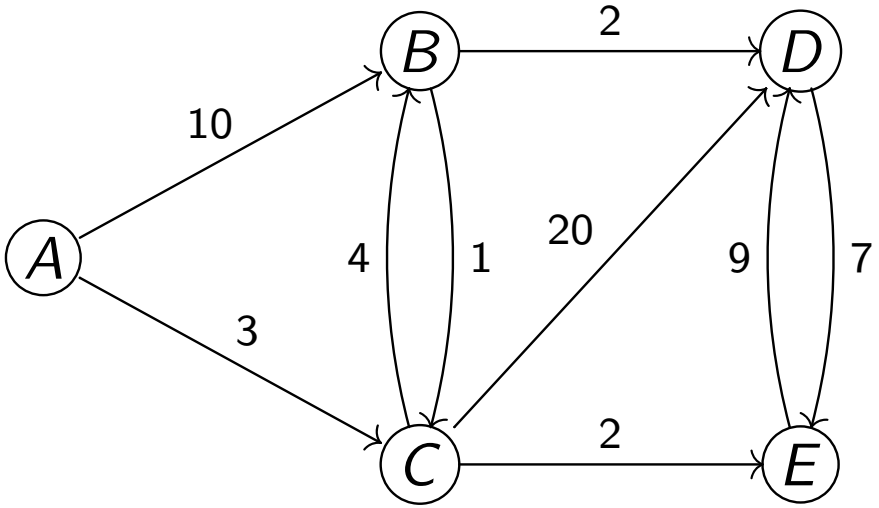
# 최단거리 알고리즘

## 연세대 미래캠퍼스 고급 알고리즘 강의 3주차

이혜아

2022년 7월 20일

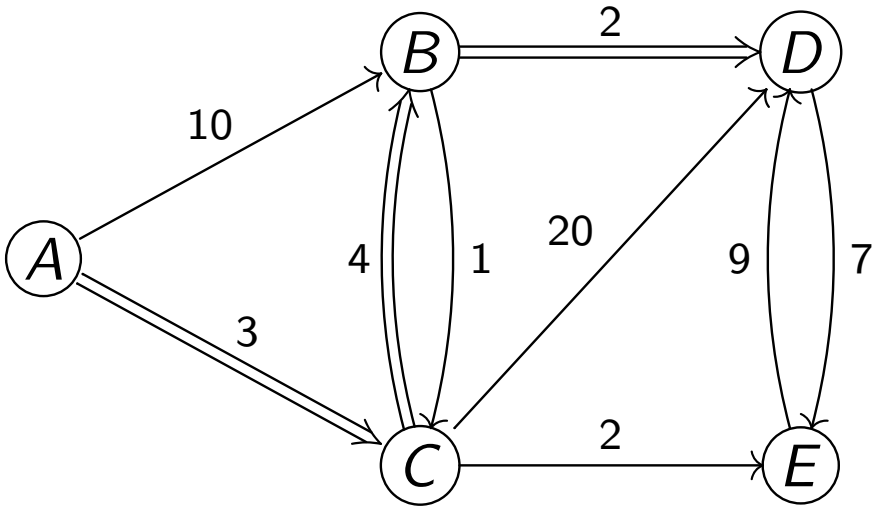
# 그래프




$V = \{A, B, C, D, E\}$
$E = \{(A, B, 10); (A, C, 3); (B, C, 1); (B, D, 2); (C, B, 4); (C, D, 20); (C, E, 2); (D, E, 7); (E, D, 9)\}$

- 그래프(*Graph*)는 정점(*Vertex*)과 간선(*Edge*)으로 이루어짐.
  - $V$: 정점의 집합. 아무것이나 될 수 있음
  - $E$: 간선의 집합. 각 원소는 두 정점을 서로 연결해야함
  - 방향이 있으면 유향(*Directed*) 그래프, 없으면 무향(*Undirected*) 그래프
  - 다른 정보가 있을 수도 있음 (위에서는 가중치)

# 경로




- 경로: 정점과 간선이 번갈아 가면서 나타나는 수열
- 정점으로 시작해서 정점으로 끝나고 $v_0, e_1, v_1, \cdots, e_m, v_m$로 표현
- 경로의 시작점: $v_0$, 도착점: $v_m$, 가중치: 간선의 가중치 합
- $s$에서 $e$까지의 최단경로: 시작점이 $s$이고 도착점이 $e$인 경로 중 가중치가 최소인 경로

# 최단거리 알고리즘

- 최단거리 알고리즘: 방향과 가중치가 있는 그래프에서 동작

| 알고리즘 | 가중치 종류 | 구하는 경로들 | 시간복잡도 |
|---|---|---|---|
| BFS | 1 | 고정된 출발점 | $O(V+E)$ |
| Floyd-Warshall | 관계 없음 | 모든 쌍 | $O(V^3)$ |
| Dijkstra | $\geq 0$ | 고정된 출발점 | $O(E \log E)$ |
| Bellman-Ford | 관계 없음 | 고정된 출발점 | $O(VE)$ |

- 방향이 없는 그래프에서 $u - v$ 간선은 $u \to v; v \to u$ 모두를 넣어주면 됨

# Floyd-Warshall

- 거리 함수가 만족하는 조건
  - $d(a, a) = 0$
  - $d(a, c) \leq d(a, b) + d(b, c)$
- 마지막 부등식은 삼각부등식이라 알려져 있음

## 연습문제

1. *(Easy)* 음수 사이클이 없는 그래프 $G$에서 $d(a, b)$를 $a$에서 $b$까지의 최단경로라고 만족하면, 거리 함수의 조건을 만족함을 증명하여라.
   - 음수 사이클: 자기 자신에서 자기 자신으로 가는 경로 중 가중치가 음수인 경로

# Floyd-Warshall

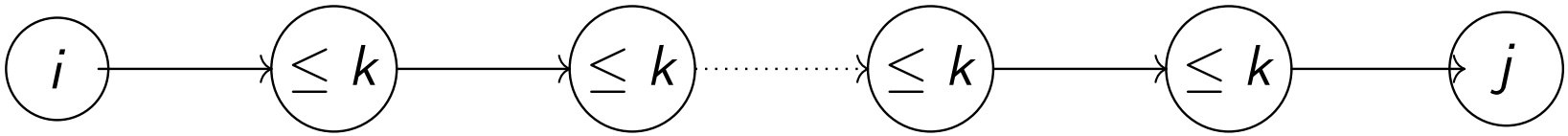
- Floyd-Warshall 알고리즘은 거리함수 갱신을 $N$번 진행

## Floyd-Warshall

1. 입력으로 다음 2차원 배열을 받음
   1. $D[a][a] = 0$
   2. $a$에서 $b$까지 가중치 $c$의 간선이 있으면 $D[a][b] = c$
   3. $a$에서 $b$까지의 간선이 없으면 $D[a][b] = \infty$
2. $k$를 1부터 $N$까지, $i$를 1부터 $N$까지 반복, $j$를 1부터 $N$까지 반복
   - $D[i][j] \leftarrow \min(D[i][j], D[i][k] + D[k][j])$
3. 최종 배열 $D[a][b]$는 $a$에서 $b$까지의 최단경로가 들어있음

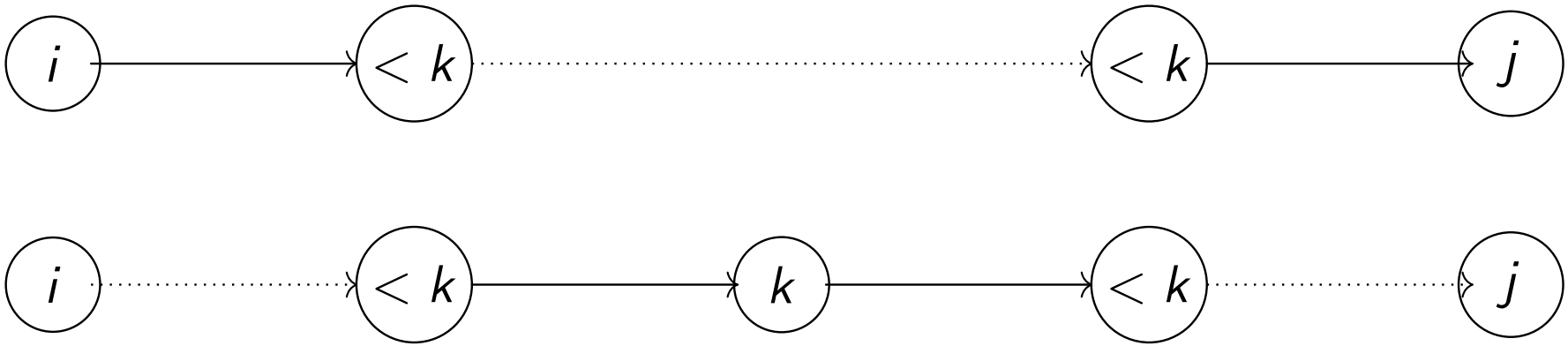
- $k$가 가장 바깥쪽 루프임을 주의. $i$와 $j$의 순서는 상관 없음
- 삼각부등식이 맞지 않는 것을 $k, i, j$순으로 업데이트 해준다고 기억하면 쉬움

# Floyd-Warshall




- $D_{k,i,j}$: $i$에서 $j$까지 경로 중 중간 정점을 $k$ 이하로 유지하는 최단거리
- $D_{N,i,j}$가 $i$와 $j$ 사이의 최단거리
- 모든 $k = 0, \cdots, N; i = 1, \cdots, N; j = 1, \cdots, N$에 대해서 구함

# Floyd-Warshall




- $D_{k,i,j}$가 $k$을 거치지 않는 경우: $D_{k,i,j} = D_{k-1,i,j}$
- $D_{k,i,j}$가 $k$을 거치는 경우: $D_{k,i,j} = D_{k-1,i,k} + D_{k-1,k,j}$
- 실제 $D_{k,i,j}$는 이 두 값중 작은 값

# Floyd-Warshall

- 코드로 구현하면 다음과 같은 식이 나옴
- $D[k][i][j] \leftarrow \min(D[k-1][i][j], D[k-1][i][k] + D[k-1][k][j])$

## 연습문제

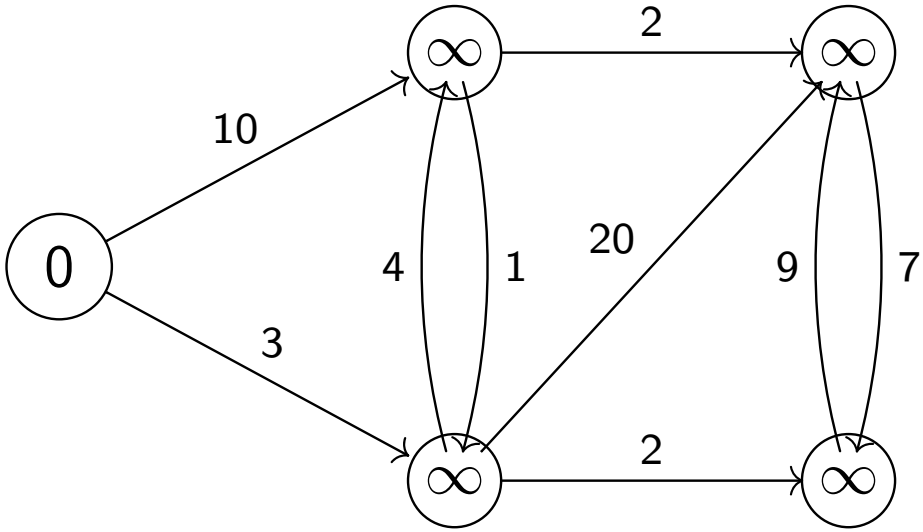
1. *(Medium)* Floyd-Warshall 코드를 보면, 3차원 점화식의 가장 첫 인덱스를 없앤 채로 구현한다. 이 구현이 올바름을 증명하여라.

## Dijkstra Algorithm

- 그래프의 모든 가중치가 0 이상일 때, $s$에서 시작해 모든 정점까지 가는 최단거리를 다음과 같은 방법으로 구할 수 있다.

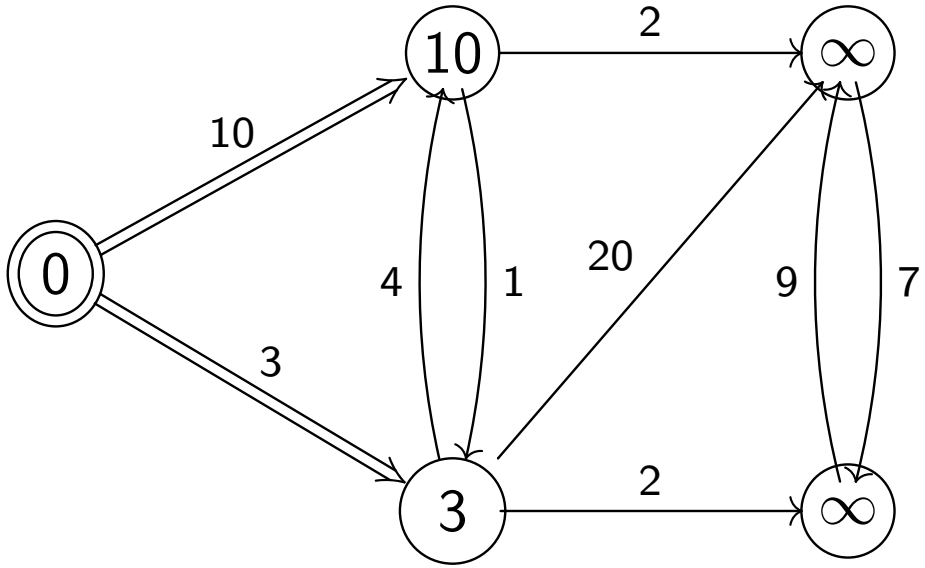
1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm



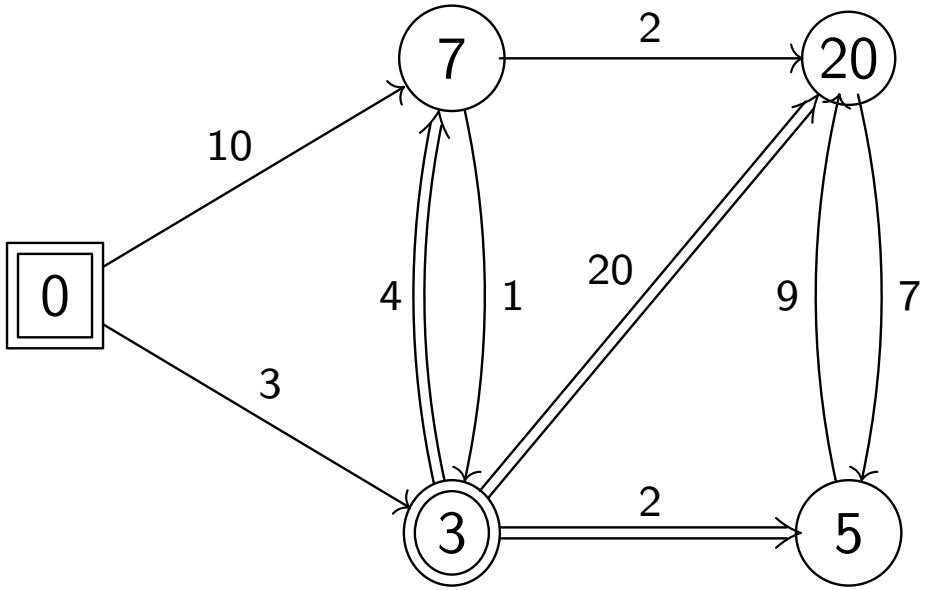

1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해 거리를 업데이트한다.

# Dijkstra Algorithm



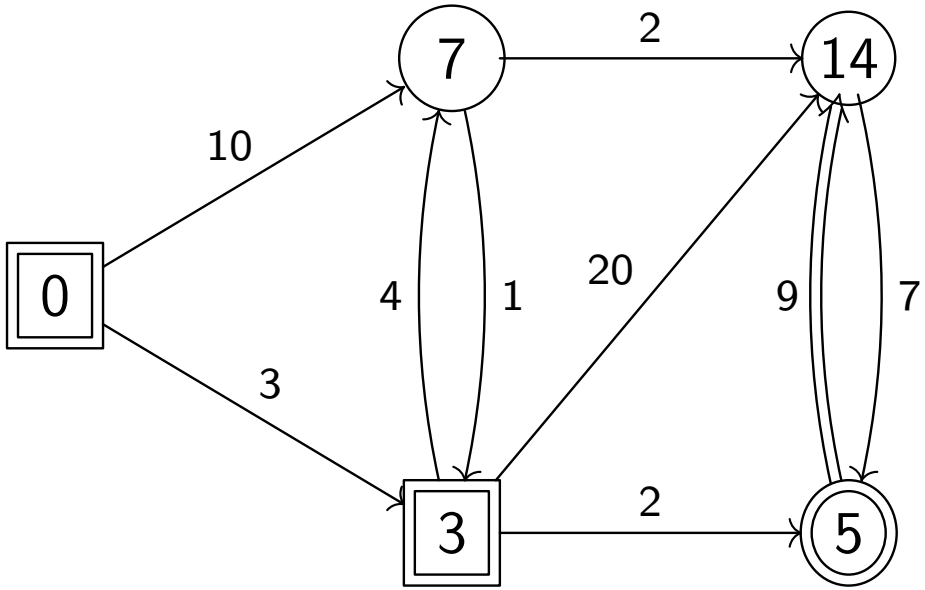

1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm



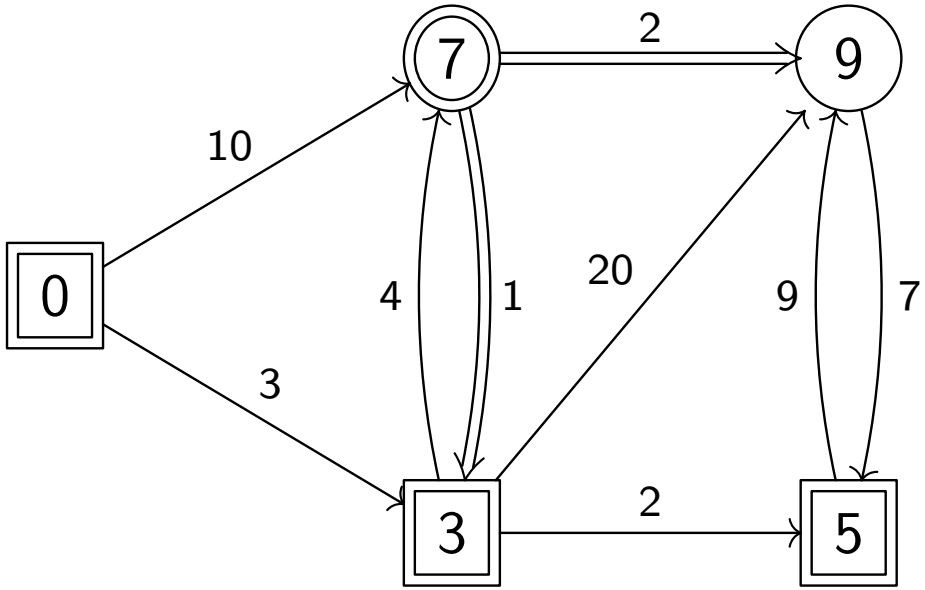

1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm



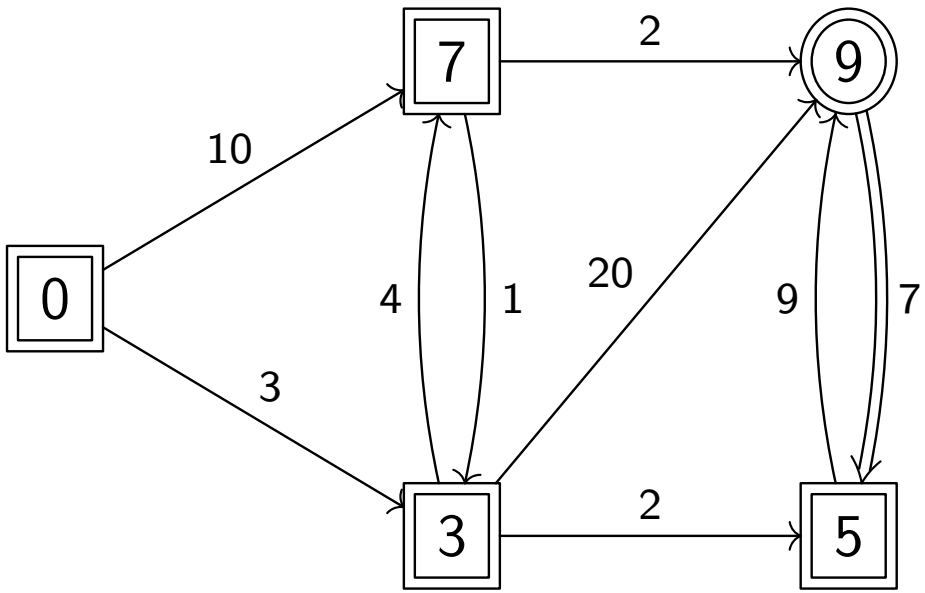

1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm



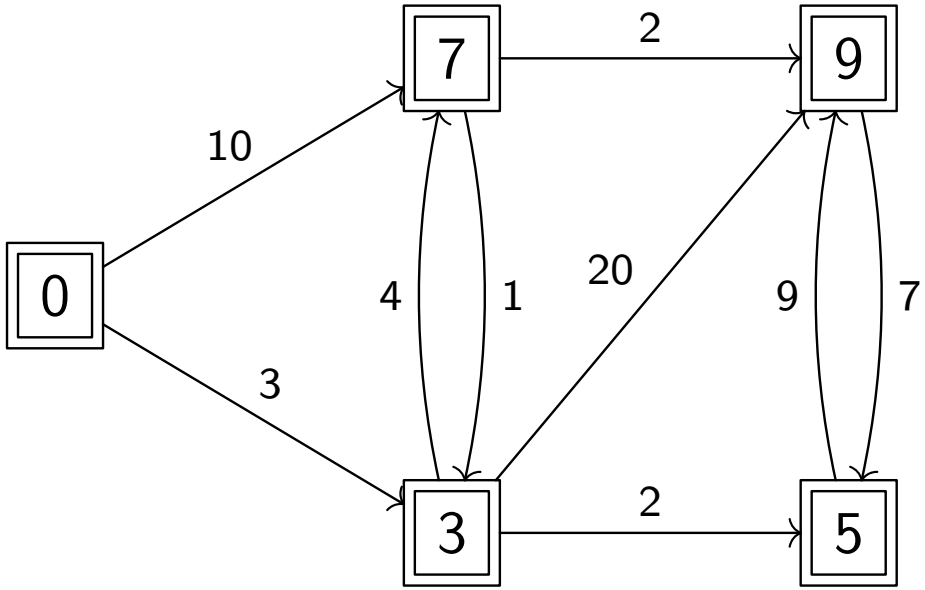

1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm




1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm




1. 처음에 거리 배열 $D$를 $\infty$로 초기화 하고, $D[s]$를 0으로 설정한다.
2. 다음을 반복한다.
   1. 거리가 **확정**되지 않은 정점 중, 거리가 제일 작은 정점의 거리를 확정한다.
   2. 해당 정점에서 갈 수 있는 정점에 대해, 거리가 더 작아지는 점이 있다면 거리를 갱신한다.

# Dijkstra Algorithm
## 증명

- 최단거리를 확정짓지 않은 점 중 거리가 제일 작은 정점을 $a$라고 하자.
- $a$로 가는 최단거리는, 기존에 최단거리가 확정된 정점만 이용함
  - 만약 최단거리가 확정되지 않는 $b$를 거쳐서 $a$에 간다고 하자.
  - $d(s,a) = d(s,b) + d(b,a)$이고, $d(s,b) \geq d(s,a); d(b,a) > 0$
  - $d(s,a) > d(s,b)$ 이므로 모순
- 수학적 귀납법 등을 사용하여, 위의 논리를 적용하면 증명할 수 있음

# Dijkstra Algorithm

- 위와 같이 구현하면 시간 복잡도는 $O(V^2 + E)$가 됨
  - 매 반복마다 가장 작은 정점을 찾는 시간이 $O(V)$가 걸림
  - 총 $O(V^2)$의 시간이 걸림
  - 모든 간선은 한 번씩 확인되기 때문에 $O(E)$ 시간이 걸림
- 빠르게 하기 위해서는 $O(V^2)$ 부분을 개선해야함
- 사용하지 않은 가장 작은 정점 $\rightarrow$ 정점을 제거
- 가장 작은 값을 찾고 삭제 $\rightarrow$ 우선순위 큐를 사용할 수 있음

# Dijkstra Algorithm

- 우선순위 큐를 올바른 방법으로 사용해야 함
- 1부터 $N$까지의 수를 모두 넣고, 비교 기준을 거리 배열로 한다.
  - ($\times$) 거리 배열의 값을 바꾸면, 힙 속성이 깨질 수 있음
- 우선순위 큐에다가 거리와 정점의 쌍을 저장함
- 거리를 갱신할 필요가 있을 때는, 우선순위 큐에 집어넣기만 함
  - 이후에 우선순위 큐에서 뺄 때 거리가 빠른 것부터 빠짐
  - 같은 정점이 여러번 보이면, 이미 작은 값을 처리했기 때문에 무시하면 됨

# Dijkstra Algorithm

## Dijkstra Algorithm ($\mathcal{O}(E \log E)$ 버전)

- 그래프의 모든 가중치가 0 이상일 때, $s$에서 시작해 모든 정점까지 가는 최단거리를 다음과 같은 방법으로 구할 수 있다.

1. 처음에 빈 우선순위 큐를 만든다. 거리와 정점의 쌍을 담으며, 거리가 작은 값이 먼저 나온다.
2. 우선순위 큐에 $(0, s)$를 담는다.
3. 우선순위 큐가 비어있지 않을 때까지 다음을 반복한다.
    1. 우선순위 큐에서 $(d, a)$를 뽑는다.
    2. 이미 $a$를 방문한 적이 있으면 해당 값을 무시하고 위로 돌아간다.
    3. $a$까지의 최단거리를 $d$로 확정해준다.
    4. 모든 $a \xrightarrow{w} b$에 대해 $(d + w, b)$를 우선순위 큐에 넣는다.

# Dijkstra Algorithm

- $a \xrightarrow{w} b$ 간선에 대해 $d(s,a) + w = d(s,b)$이면 $s$에서 $b$까지 가는 최단경로 중 $a \xrightarrow{w} b$를 이용하는 최단경로가 존재함
- $s$에서 $b$까지 가는 최단경로인 경로를 찾기 위해서는, $b$에서 모든 $a \xrightarrow{w} b$를 보면서 해당 간선이 최단경로에 이용되는지 확인
- 이제 $s$에서 $b$까지 가는 최단경로는 $s$에서 $a$로 가고, $a \xrightarrow{w} b$를 이용함
- 같은 방법으로 $s$에서 $a$까지 가는 최단경로를 찾으면 됨 (반복문 사용)

# 최단거리 알고리즘의 응용

- 상태를 바꾸는데 특정 비용이 들고, 초기 상태에서 원하는 상태까지 바꾸는 최소 비용을 묻는 문제가 있다고 하자.
- 이는 상태를 정점, 상태를 바꾸는 연산을 간선으로 하는 그래프로 만들면 최단경로 문제로 바뀜
- 문제에 따라, 어떤 것을 상태로 잡는지가 매우 중요

# 최단거리 알고리즘의 응용

## 연습문제

1. *(Hard)* $s$에서 $t$까지 가는 경로의 비용을 계산할 때, 경로에 속한 간선 중 비용이 제일 높은 $K$개를 제외한 나머지 간선 가중치 합이라고 하자. 그래프와 $K, s, t$가 주어졌을 때 최단경로를 구하여라.
   - 문제: `https://www.acmicpc.net/problem/1162`
2. *(Hard)* 모든 자리수가 0과 1로 이루어진 $N$의 배수 중 가장 작은 수를 구하여라.
   - 예시: 17이 주어지면 $11101 = 17 \times 653$
   - 문제: `https://www.acmicpc.net/problem/8112`
3. *(Very Hard)* $a_1, \cdots, a_n$이 주어질 때, $M$이 $a_i$들의 합으로 표현되는지 구하여라.
   - 예시: $a_1 = 4, a_2 = 7$; $15 = 4 + 4 + 7$로 표현 가능, 17은 표현할 수 없음
   - `https://www.acmicpc.net/problem/8008`