ALGORITHM
알고리즘

Lecture 10

그래프 (3)

컴퓨터과학과 | 이관용 교수

한국방송통신대학교

학습목차

01. 최단 경로: 벨만-포드 알고리즘

벨만-포드 알고리즘?

단일 출발점 최단 경로를 구하는 알고리즘

- 음의 가중치를 갖는 간선이 존재하는 경우에도 처리 가능
  - ✓ 음의 사이클이 없는 경우에 한함

G=(V, E)에서 |V|=n일 때 단계적으로 최단 경로를 구해 나가는 방법

- 간선을 최대 1개 사용하는 최단 경로
- 간선을 최대 2개 사용하는 최단 경로

……

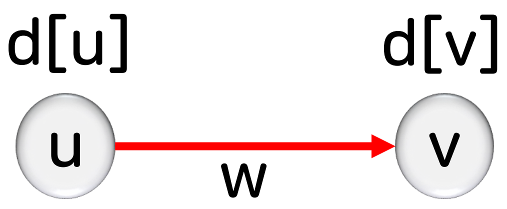
- 간선을 최대 (n-1)개 사용하는 최단 경로

벨만-포드 알고리즘

수행 과정

- 초기화
  - ✓ 출발점 s의 거리 d[s]=0, 나머지 모든 정점 v의 거리 d[v]=∞
- for i ← 1 to |V|-1
  - ✓ 모든 간선을 한 번씩 조사하면서 거리값 조정 여부를 결정

    d[v] ← min{ 기존 거리, 간선 ⟨u, v⟩를 지나는 경우의 거리 }



d[v] = min{ d[v], d[u]+w }

전 단계에서 거리값 d[ ]의 조정이 발생한 정점에 부수된 간선만 조사

벨만-포드 알고리즘

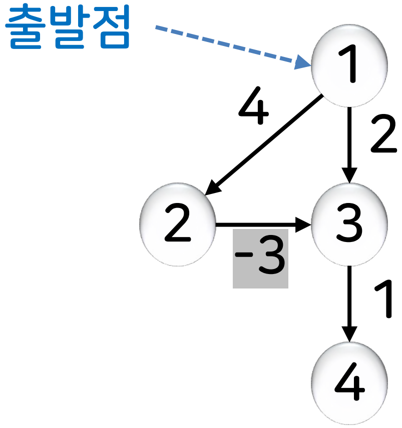

```
Bellman_Ford (G, s) {
  for ( 모든 정점 v∈V ) {
    d[v] = ∞;
    prev[v] = NULL;
  }
  d[s] = 0;
  for (i=1; i<n; i++) {
    for ( 모든 간선 (u, v)∈E )
      if ( d[v] > d[u] + W(u, v) ) {
        d[v] = d[u] + W(u, v);
        prev[v] = u;
      }
  }
  return (d[ ], prev[ ]);
}
```

입력: G=(V,E), s : 시작 정점

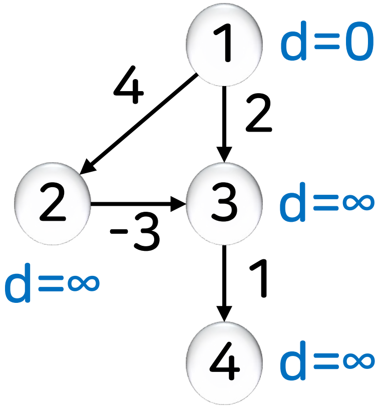
출력: d[] : s로부터 다른 모든 정점으로의 최단 경로의 길이

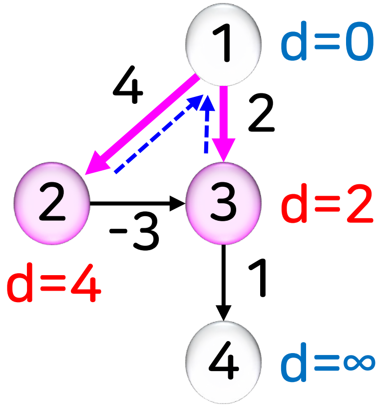
prev[] : 최단 경로를 만드는 선행 정점

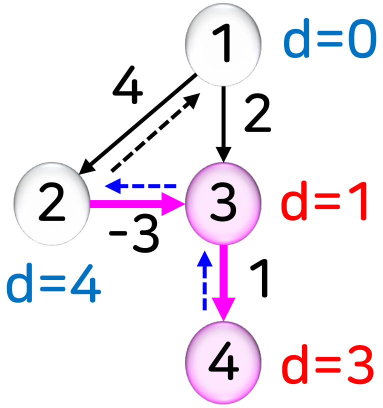
벨만-포드 알고리즘_예_1

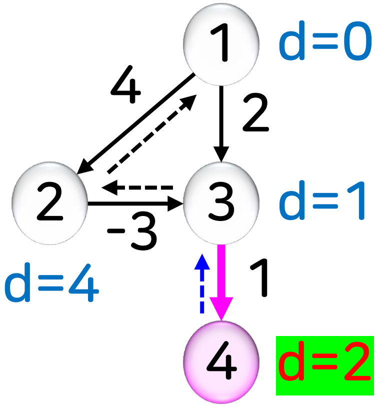


초기화







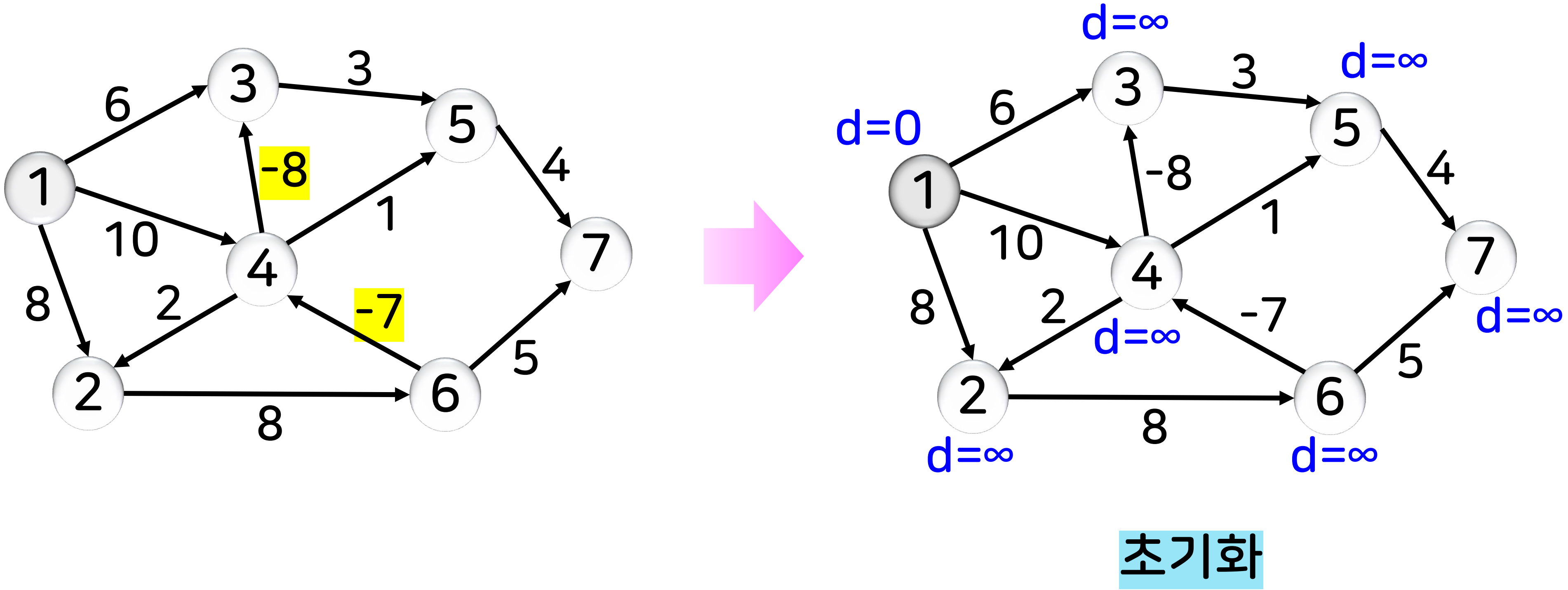


간선 1개를 사용한 최단 경로

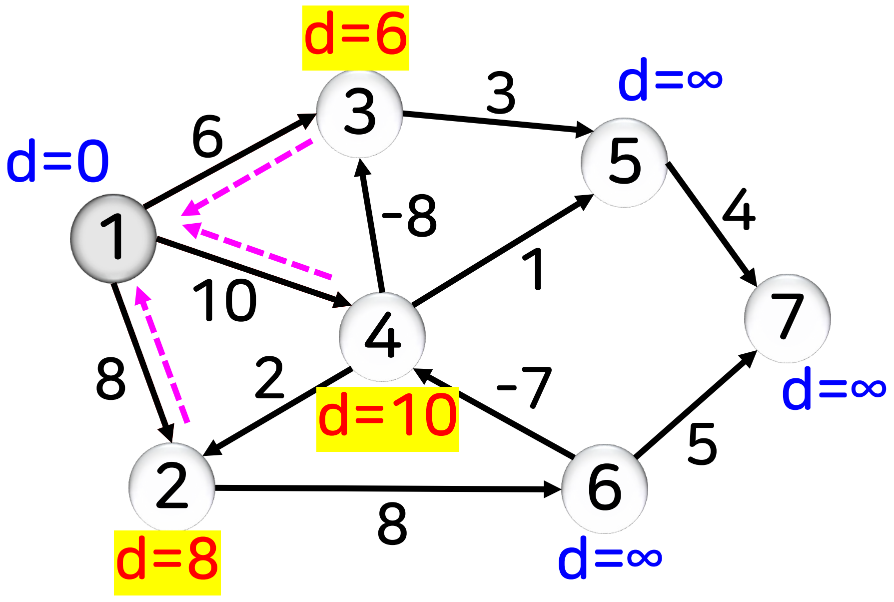
간선 2개를 사용한 최단 경로

간선 3개를 사용한 최단 경로

벨만-포드 알고리즘_예_2

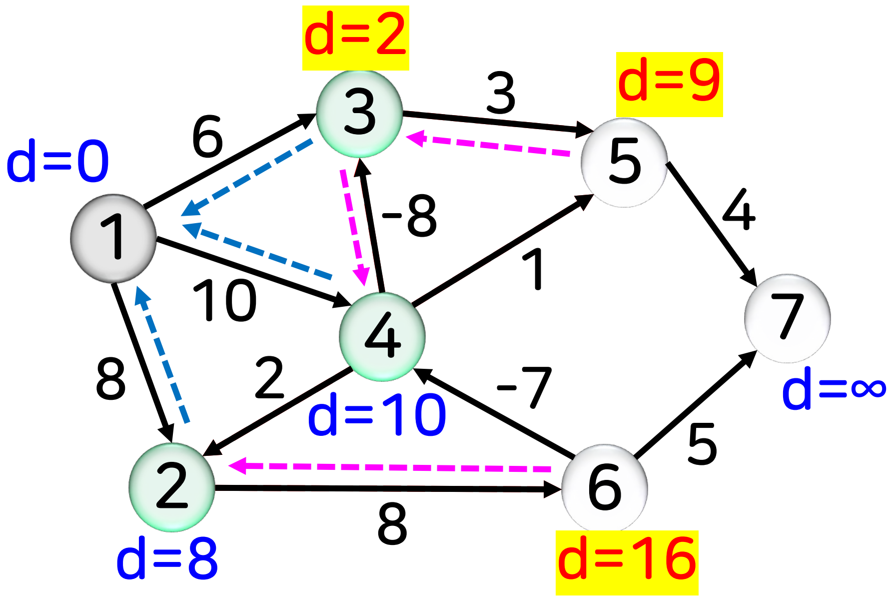


벨만-포드 알고리즘_예_2



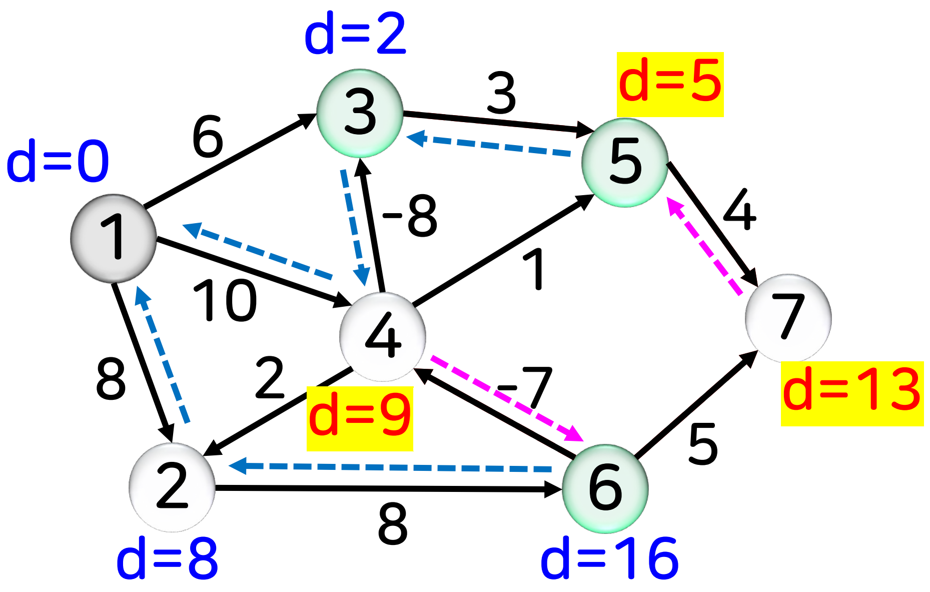
간선 1개를 사용한 경우

벨만-포드 알고리즘_예_2



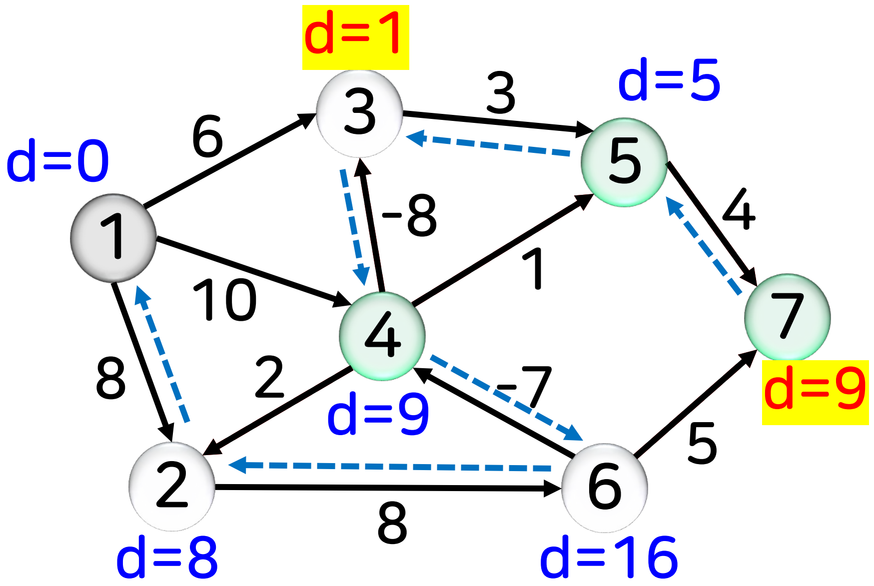
간선 2개를 사용한 경우

벨만-포드 알고리즘_예_2



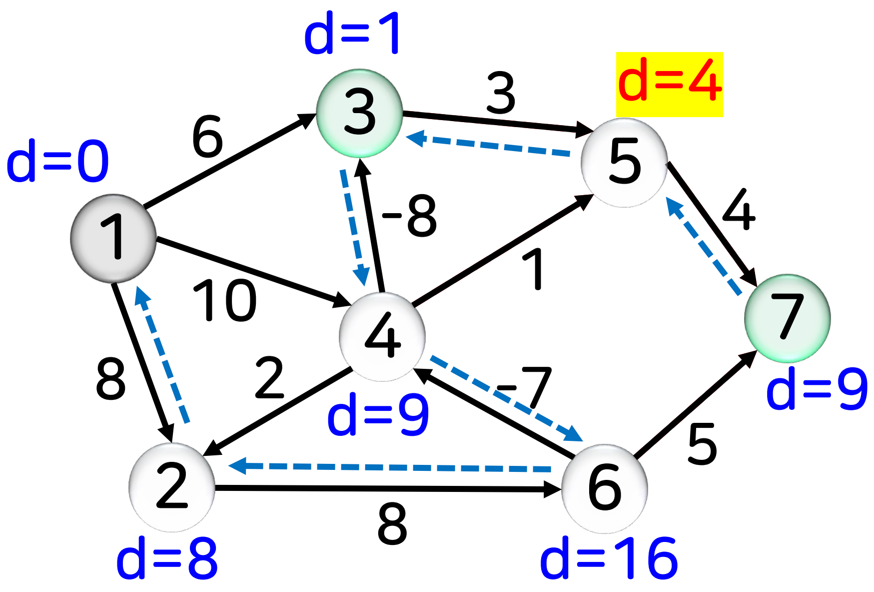
간선 3개를 사용한 경우

벨만-포드 알고리즘_예_2



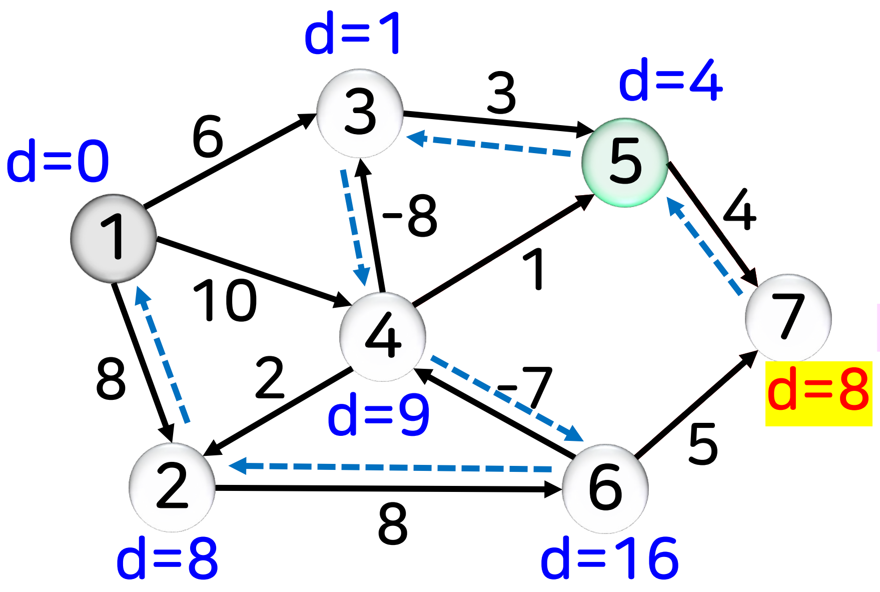
간선 4개를 사용한 경우

벨만-포드 알고리즘_예_2

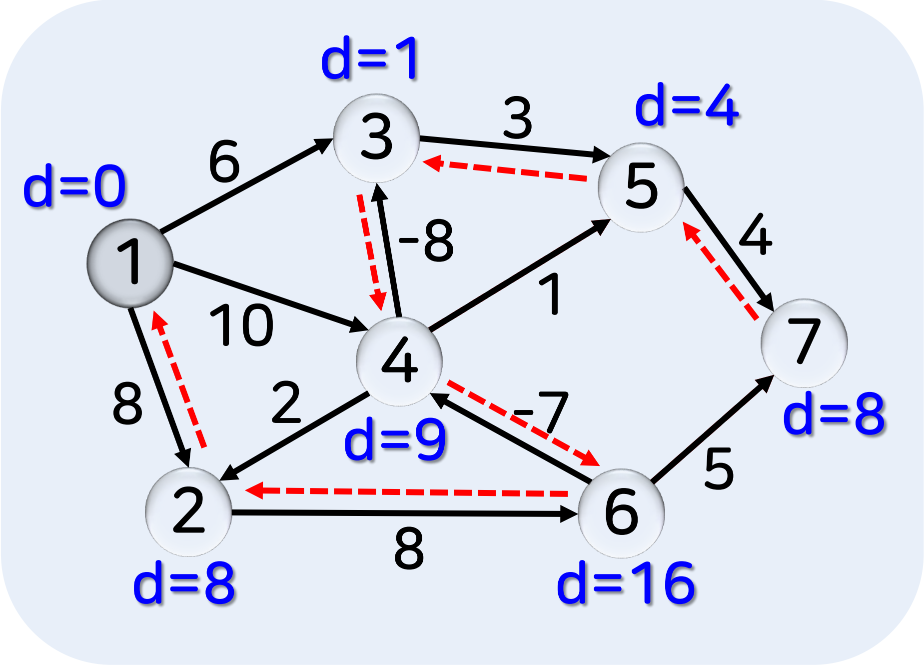


간선 5개를 사용한 경우

벨만-포드 알고리즘_예_2



간선 6개를 사용한 경우



최종 결과

성능과 특징

01

```
Bellman_Ford (G, s) {
  for ( 모든 정점 v∈V ) {
    d[v] = ∞;
    prev[v] = NULL;
  }
  d[s] = 0;
  for (i=1; i<n; i++) {
    for ( 모든 간선 (u, v)∈E )
      if ( d[v] > d[u] + W(u, v) ) {
        d[v] = d[u] + W(u, v);
        prev[v] = u;
      }
  }
  return (d[ ], prev[ ]);
}
```

$O(|V|)$

$O(|V|)$

$O(|E|)$

→ $O(|V||E|)$

성능과 특징

음의 가중치를 갖는 간선이 있는 경우

- 데이크스트라 알고리즘 적용 불가
- 벨만-포드 알고리즘 적용 가능
  - ✓ 단, 음의 사이클이 존재하면 적용 불가

음의 가중치를 갖는 간선이 없는 경우

- 데이크스트라 알고리즘 → $O((|V|+|E|) \log|V|)$ ⇒ 바람직
- 벨만-포드 알고리즘 → $O(|V||E|)$

02. 최단 경로: 플로이드 알고리즘

플로이드 알고리즘?

모든 쌍 최단 경로 알고리즘

- "Floyd-Warshall 알고리즘"
- 가정 → 경로의 길이가 음인 사이클이 존재하지 않음
- 적용 기법 → 동적 프로그래밍 방법
- 경유할 수 있는 정점의 범위가 1인 경로부터 시작해서 |V|인 경로까지
  하나씩 정점의 범위를 늘려 가면서 모든 정점 간의 최단 경로를 한꺼번에 구함

플로이드 알고리즘?

인접 행렬 표현 활용

$$D^{(k)} = \left(d_{ij}^{(k)}\right) \quad k = 0,1,\cdots,|V|$$

정점 번호가 k 이하인 정점만을 경유하는
정점 i에서 j까지의 최단 경로의 길이

$$D^{(0)} = \left(d_{ij}^{(0)} = w(i,j)\right) \rightarrow D^{(1)} \rightarrow D^{(2)} \rightarrow \cdots \rightarrow D^{(|V|)}$$

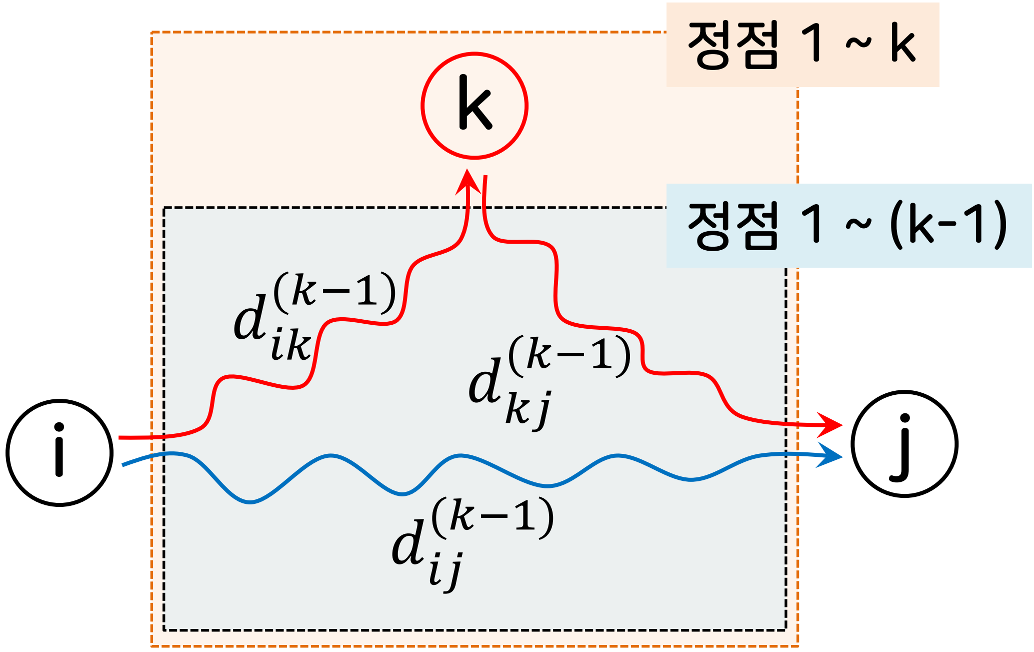
최종 결과

점화식

$$d_{ij}^{(k)} = \min\left(\underbrace{d_{ij}^{(k-1)}}_{\text{기존 거리}},\ \underbrace{d_{ik}^{(k-1)} + d_{kj}^{(k-1)}}_{\text{정점 k를 경유하는 경우의 거리}} \right)$$

플로이드 알고리즘?

$d_{ij}^{(k-1)}$과 $d_{ij}^{(k)}$의 관계



$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right)$$

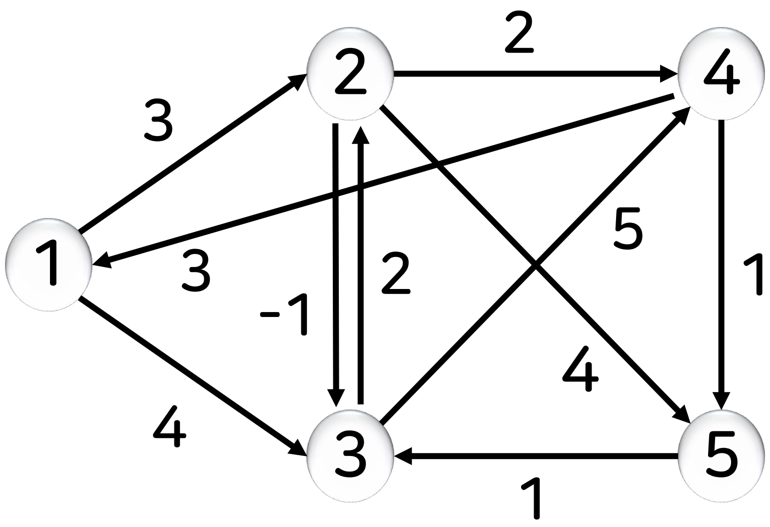
플로이드 알고리즘

입력: G=(V,E), 인접 행렬 A[1..n][1..n]
출력: D[][] : 모든 정점 쌍 간의 최단 경로의 길이

```
Floyd (G)
{
  for (i=1; i<=n; i++)          // D^(0) = ( d_ij^(0) = w_ij )
    for (j=1; j<=n; j++)
      D[i][j] = A[i][j];
  for (k=1; k<=n; k++)          // D^(k): 경유하는 정점 k
    for (i=1; i<=n; i++)        // D_ij
      for (j=1; j<=n; j++)
        if (D[i][j] > D[i][k] + D[k][j] )  // 경유하는 경우
          D[i][j] = D[i][k] + D[k][j];
  return (D[ ][ ]);
}
```

플로이드 알고리즘_예_1

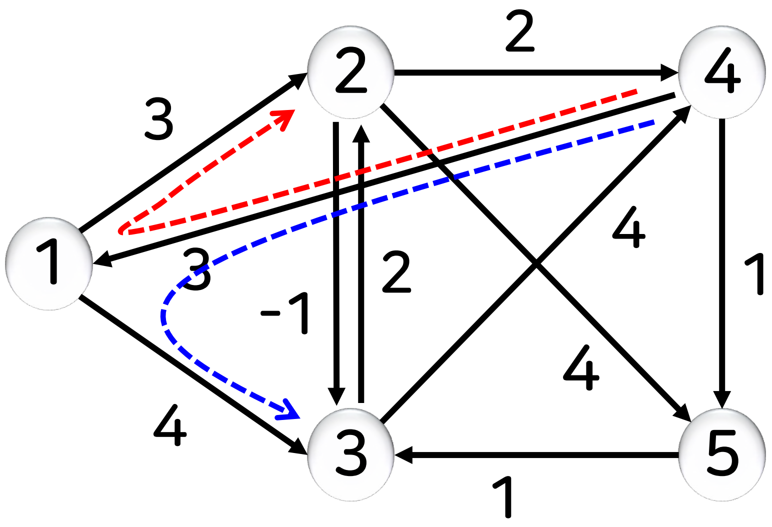
$D^{(0)}$ → 그래프를 인접행렬로 표현/초기화



$$D^{(0)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & 4 & \infty & \infty \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 5 & \infty \\ 3 & \infty & \infty & 0 & 1 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix} \end{array}$$

플로이드 알고리즘_예_1

D(1)



$$D^{(0)} = \begin{bmatrix} 0 & 3 & 4 & \infty & \infty \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 5 & \infty \\ 3 & \infty & \infty & 0 & 1 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

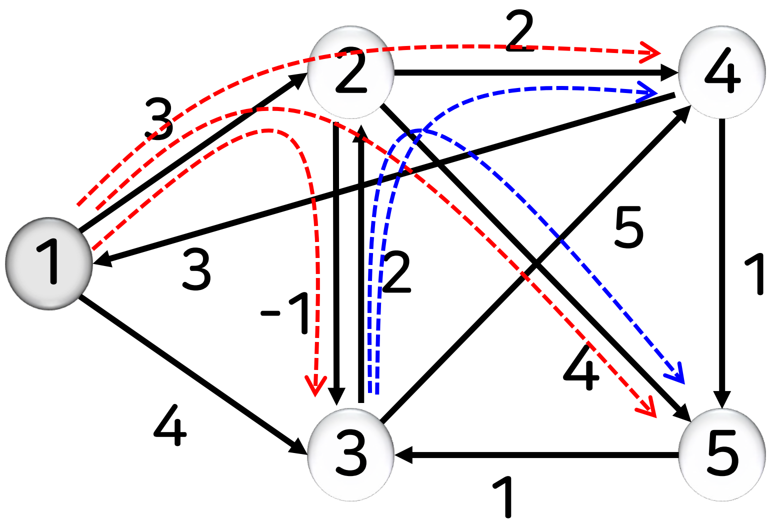
$$d_{42}^{(1)} = \min\left(d_{42}^{(0)},\ d_{41}^{(0)} + d_{12}^{(0)}\right) = \min(\infty, 3+3) = 6$$

$$d_{43}^{(1)} = \min\left(d_{43}^{(0)},\ d_{41}^{(0)} + d_{13}^{(0)}\right) = \min(\infty, 3+4) = 7$$

$$D^{(1)} = \begin{bmatrix} 0 & 3 & 4 & \infty & \infty \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 5 & \infty \\ 3 & 6 & 7 & 0 & 1 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

플로이드 알고리즘_예_1

$D^{(2)}$



$$D^{(1)} = \begin{bmatrix} 0 & 3 & 4 & \infty & \infty \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 5 & \infty \\ 3 & 6 & 7 & 0 & 1 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

$$d_{1j}^{(2)} = \min\left(d_{1j}^{(1)},\ d_{12}^{(1)} + d_{2j}^{(1)}\right) \quad j = 3,4,5$$

$d_{1j}^{(1)}$	$d_{12}^{(1)} + d_{2j}^{(1)}$
4	3-1
∞	3+2
∞	3+4

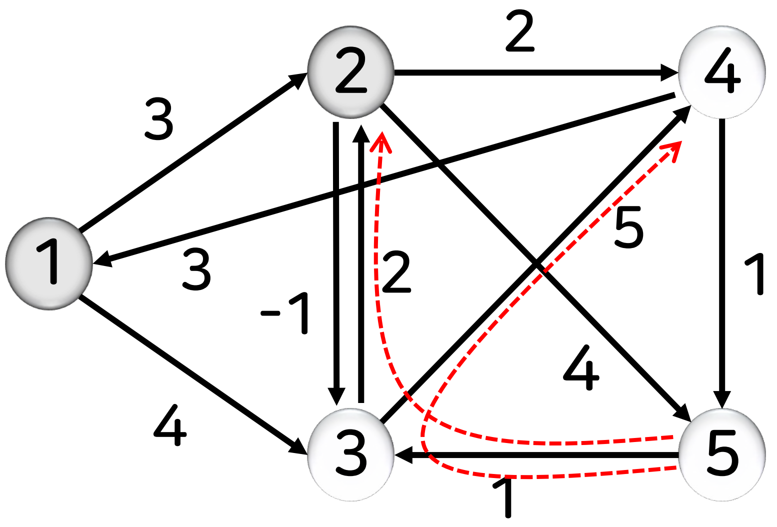
$$d_{3j}^{(2)} = \min\left(d_{3j}^{(1)},\ d_{32}^{(1)} + d_{2j}^{(1)}\right) \quad j = 4,5$$

$d_{3j}^{(1)}$	$d_{32}^{(1)} + d_{2j}^{(1)}$
5	2+2
∞	2+4

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 7 \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 4 & 6 \\ 3 & 6 & 7 & 0 & 1 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

플로이드 알고리즘_예_1

$D^{(3)}$



$$D^{(2)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 7 \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 4 & 6 \\ 3 & 6 & 7 & 0 & 1 \\ \infty & \infty & 1 & \infty & 0 \end{bmatrix}$$

$d_{12}^{(3)}$ $d_{14}^{(3)}$ $d_{24}^{(3)}$ → 거리값 변화 없음

$$d_{52}^{(3)} = \min\left(d_{52}^{(2)},\ d_{53}^{(2)} + d_{32}^{(2)}\right)$$

∞ 1+2

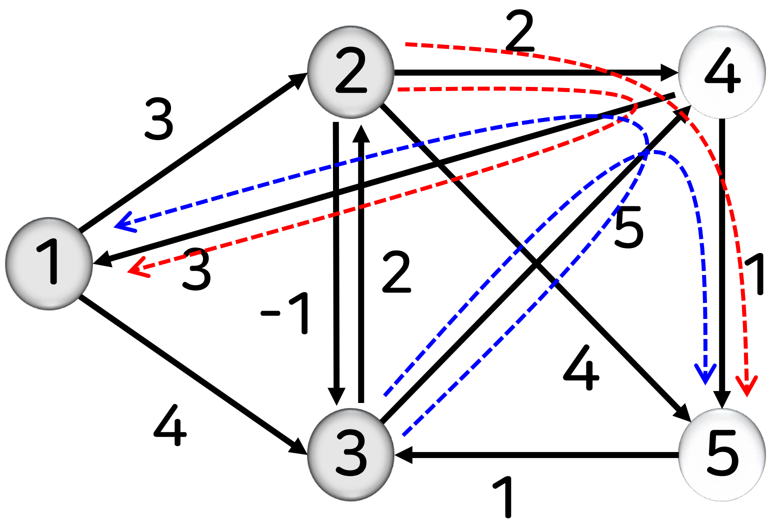
$$d_{54}^{(3)} = \min\left(d_{54}^{(2)},\ d_{53}^{(2)} + d_{34}^{(2)}\right)$$

∞ 1+4

$$D^{(3)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 7 \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 4 & 6 \\ 3 & 6 & 7 & 0 & 1 \\ \infty & 3 & 1 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘_예_1

$D^{(4)}$



$$d^{(4)}_{21} = \min\left(\underset{\infty}{d^{(3)}_{21}},\ \underset{2+3}{d^{(3)}_{24}+d^{(3)}_{41}}\right)$$

$$d^{(4)}_{25} = \min\left(\underset{4}{d^{(3)}_{25}},\ \underset{2+1}{d^{(3)}_{24}+d^{(3)}_{45}}\right)$$

$$d^{(4)}_{31} = \min\left(\underset{\infty}{d^{(3)}_{31}},\ \underset{4+3}{d^{(3)}_{34}+d^{(3)}_{41}}\right)$$

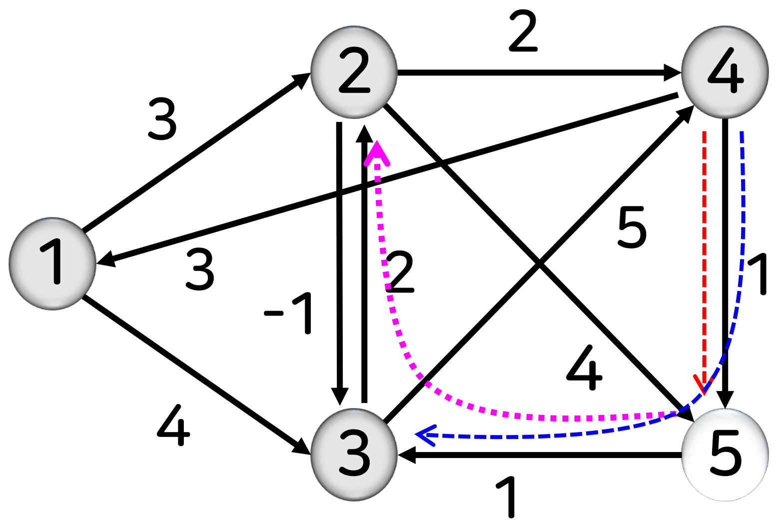
$$d^{(4)}_{35} = \min\left(\underset{6}{d^{(3)}_{35}},\ \underset{4+1}{d^{(3)}_{34}+d^{(3)}_{45}}\right)$$

$$D^{(3)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 7 \\ \infty & 0 & -1 & 2 & 4 \\ \infty & 2 & 0 & 4 & 6 \\ 3 & 6 & 7 & 0 & 1 \\ \infty & 3 & 1 & 5 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 6 \\ 5 & 0 & -1 & 2 & 3 \\ 7 & 2 & 0 & 4 & 5 \\ 3 & 6 & 7 & 0 & 1 \\ 8 & 3 & 1 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘_예_1

D(5)



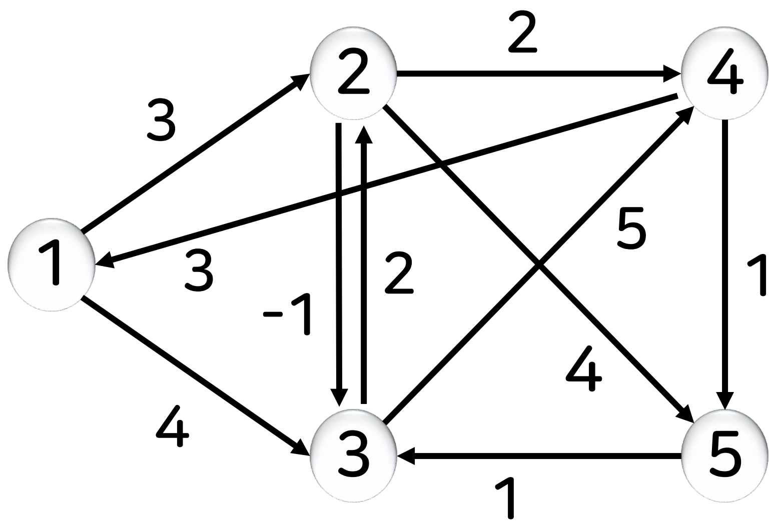
$$D^{(4)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 6 \\ 5 & 0 & -1 & 2 & 3 \\ 7 & 2 & 0 & 4 & 5 \\ 3 & 6 & 7 & 0 & 1 \\ 8 & 3 & 1 & 5 & 0 \end{bmatrix}$$

$$d_{42}^{(5)} = \min\left(d_{42}^{(4)},\ d_{45}^{(4)} + d_{52}^{(4)}\right) = \min(6, 1+3) = 4$$

$$d_{43}^{(5)} = \min\left(d_{43}^{(4)},\ d_{45}^{(4)} + d_{53}^{(4)}\right) = \min(7, 1+1) = 2$$

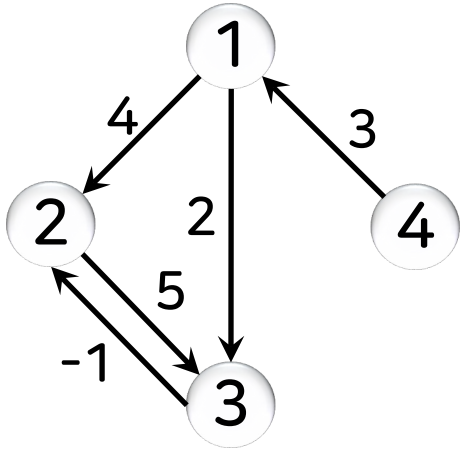
$$D^{(5)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 6 \\ 5 & 0 & -1 & 2 & 3 \\ 7 & 2 & 0 & 4 & 5 \\ 3 & 4 & 2 & 0 & 1 \\ 8 & 3 & 1 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘_예_1



$$D^{(0)} \rightarrow D^{(1)} \rightarrow D^{(2)} \rightarrow D^{(3)} \rightarrow D^{(4)} \longrightarrow D^{(5)} = \begin{bmatrix} 0 & 3 & 2 & 5 & 6 \\ 5 & 0 & -1 & 2 & 3 \\ 7 & 2 & 0 & 4 & 5 \\ 3 & 4 & 2 & 0 & 1 \\ 8 & 3 & 1 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘_예_2



$$D^{(0)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & \infty & \infty & 0 \end{bmatrix} \qquad D^{(1)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 7 & 5 & 0 \end{bmatrix}$$

$$d_{42}^{(1)} = \min\left(d_{42}^{(0)},\ d_{41}^{(0)} + d_{12}^{(0)}\right) = \min(\infty, 3+4) = 7$$

$$d_{43}^{(1)} = \min\left(d_{43}^{(0)},\ d_{41}^{(0)} + d_{13}^{(0)}\right) = \min(\infty, 3+2) = 5$$

$$D^{(2)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 7 & 5 & 0 \end{bmatrix} \qquad D^{(3)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix} \qquad D^{(4)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

$$d_{12}^{(3)} = \min\left(d_{12}^{(2)},\ d_{13}^{(2)} + d_{32}^{(2)}\right) = \min(4, 2-1) = 1$$

$$d_{42}^{(3)} = \min\left(d_{42}^{(2)},\ d_{43}^{(2)} + d_{32}^{(2)}\right) = \min(7, 5-1) = 4$$

성능과 특징


```
Floyd (G)
{
  for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
      D[i][j] = A[i][j];

  for (k=1; k<=n; k++)
    for (i=1; i<=n; i++)
      for (j=1; j<=n; j++)
        if (D[i][j] > D[i][k] + D[k][j] )
          D[i][j] = D[i][k] + D[k][j];

  return (D[ ][ ]);
}
```

$O(n^2)$

$O(n^3)$

$O(n^3)$ n=|V|

성능과 특징

동적 프로그래밍 방법을 적용한 알고리즘

$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right)$$

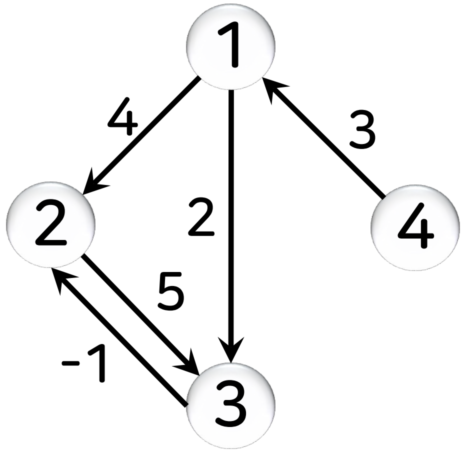
데이크스트라 알고리즘으로 모든 쌍 최단 경로를 구할 수 있음

- 각 정점에 대해서 반복적으로 적용해서 해결 가능 → $O(|V|^3)$
- 플로이드 알고리즘이 더 간단하므로 빠르게 수행
  - ✓ $|E| \ll |V|^2$인 경우에는 데이크스트라 알고리즘의 반복 적용이 더 효과적

성능과 특징

P[1..n][1..n]을 활용하면 최단 경로 자체를 구할 수 있음

- P[i][j] = k if $d_{ij}^{(k-1)} > \left(d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$ (정점 k를 경유하는 경우)



P[i][j]

0	3	0	0
0	0	0	0
0	0	0	0
0	3	1	0

$$D^{(4)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

P[4][2]=3

정점4 → 정점3 → 정점2

P[4][3]=1   P[3][2]=0

정점4 → 정점1 → 정점3

P[4][1]=0   P[1][3]=0

정점4 → 정점1 → 정점3 → 정점2

03. 네트워크 플로 문제:
포드-풀커슨 알고리즘

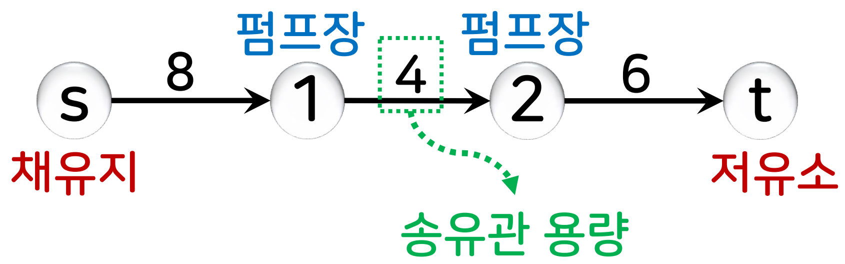
네트워크 플로 문제?

주어진 네트워크에 대해서 플로를 최대로 하는 값을 찾는 문제

- 소스에서 싱크로 보낼 수 있는 플로 값을 최대로 하는 문제 → "최대 플로 문제"

네크워크 N=(V, E, s, t, c)

- 방향 그래프 G=(V, E)
- s → 소스 (시작점, 진입차수가 0인 정점)
- t → 싱크 (도착점, 진출차수가 0인 정점)
- c → 간선의 가중치 → 간선의 용량capacity
  - ✓ c(u, v) → 간선 ⟨u, v⟩를 통해 보낼 수 있는 최대의 양/값



네트워크 플로 문제?

플로 f: E → R⁺

- 간선의 용량 중에서 실제 사용하고 있는 양/값
- 용량 제약 조건 → 모든 ⟨u, v⟩∈E에 대해서 0≤ f(u, v) ≤ c(u, v)
  - ✓ 임의의 간선의 플로는 항상 0보다 크거나 같고, 해당 간선의 용량을 초과하지 못함
- 플로 보존 제약 조건 → 모든 v∈V-{s, t}에 대하여 $\sum_{u} f(u,v) = \sum_{w} f(v,w)$
  - ✓ 소스/싱크를 제외한 모든 정점에 대해서 어떤 정점으로 들어오는 양과 나가는 양은 동일

플로 값(총 플로) F

$$F = \sum_{v \in V} f(s,v) = \sum_{w \in V} f(w,t)$$

- ✓ 소스에서 나가는 플로의 합, 싱크로 들어오는 플로의 합

포드-풀커슨 알고리즘?

최초로 제시된 가장 기초적인 해결 방법 by Ford & Fulkerson

- 단순히 플로 값을 증가시킬 수 있는 모든 경우의 수를 탐색해서 적용
- 종료가 보장되지 않음 → 에드몬즈-카프 알고리즘으로 발전

모든 간선의 플로를 0으로 둔 상태에서 시작해서 증가 경로가 더 이상 존재하지 않을 때까지 반복적으로 경로를 찾아서 최대 플로 값을 구하는 방법

포드-풀커슨 알고리즘

증가 경로 augmenting path

- 소스에서 싱크까지 더 많은 플로를 보낼 수 있는 경로
- 경로상의 간선은 네트워크 N의 간선 방향과 일치하지 않을 수 있음
  - ✓ 순방향 forward 간선 → 간선 ⟨u, v⟩가 N의 간선의 방향과 같은 경우
  - ✓ 역방향 backward 간선 → 간선 ⟨u, v⟩가 N의 간선의 방향과 다른 경우

    → ⟨v, u⟩∈E와 방향이 반대인 간선 → ⟨u, v⟩∉E인 가상의 간선

    → $c(u, v) = 0,\ \ f(u, v) = -f(v, u)$

    → 남아 있는 모든 가능한 경로를 더 찾아낼 수 있게 하는 포드-풀커슨 방법의 핵심

포드-풀커슨 알고리즘

잔여 용량 r(u, v)

- 간선 ⟨u, v⟩에서 추가로 플로를 증가시킬 수 있는 여유 용량
  - ✓ 순방향 간선 → r(u, v) = c(u, v) – f(u, v)
    → 증가시킬 수 있는 양/값
  - ✓ 역방향 간선 → r(v, u) = c(v, u) – f(v, u) = c(v, u) – (–f(u, v)) = 0 + f(u, v) = f(u, v)
    → 원래 간선에 부여된 플로를 줄일 수 있는 양/값

증가 경로의 여유량 Δ

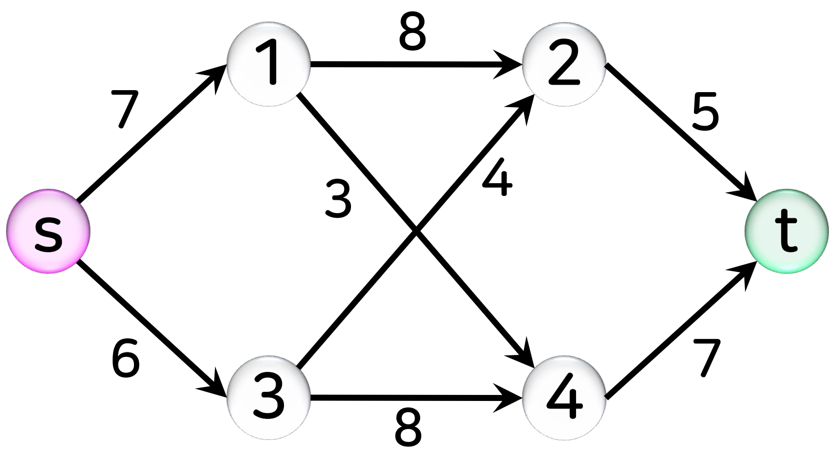
- Δ = min{ r(u, v) | 경로상의 존재하는 간선 ⟨u, v⟩ }
  - ✓ 경로에 포함된 모든 간선의 잔여 용량 중 최소값
    → 해당 경로를 사용해서 증가시킬 수 있는 플로 값

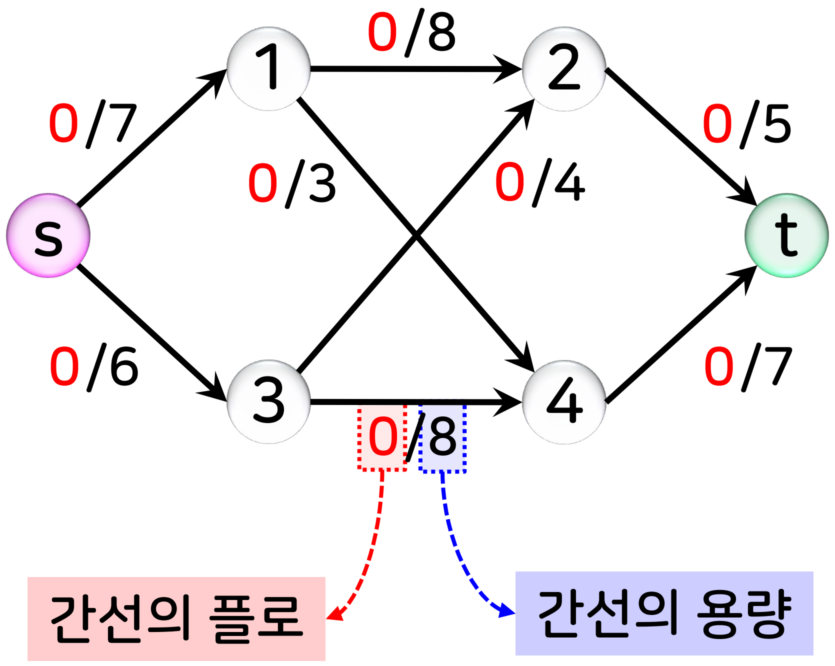
포드-풀커슨 알고리즘


```
Ford_Fulkerson ( N )
{
  모든 간선의 플로 f를 0으로 초기화;
  while ( 증가 경로 p가 존재 ) {
    Δ = min{ r(u, v) | p에 존재하는 <u, v> }
    for ( 증가 경로 p상의 모든 간선 <u, v> )
      if ( <u, v>가 순방향 간선 ) f(u, v) += Δ;     // 플로 증가
      else f(u, v) -= Δ;     // 역방향 간선 → 플로 감소
  }
  F = Σ_{w∈V} f(w, t);        // 싱크로 들어오는 모든 간선의 플로의 합
  return (F);
}
```

포드-풀커슨 알고리즘_예_1

초기화

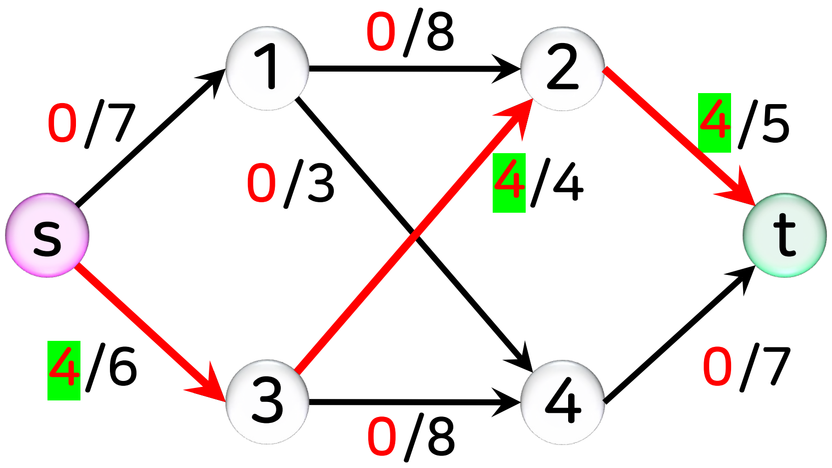




증가 경로를 찾고 해당 경로의 여유량으로
모든 간선의 플로를 조정하는 과정을 반복

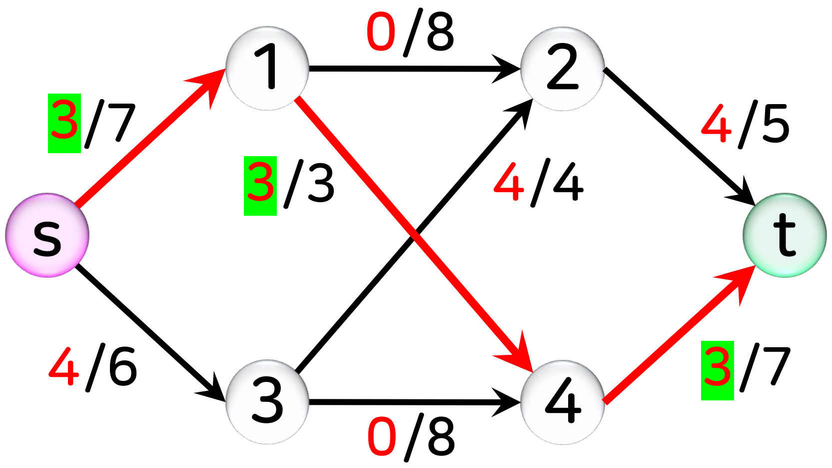
포드-풀커슨 알고리즘_예_1

증가 경로 s-3-2-t → 증가 경로의 여유량 = <3, 2>의 잔여 용량 = 4



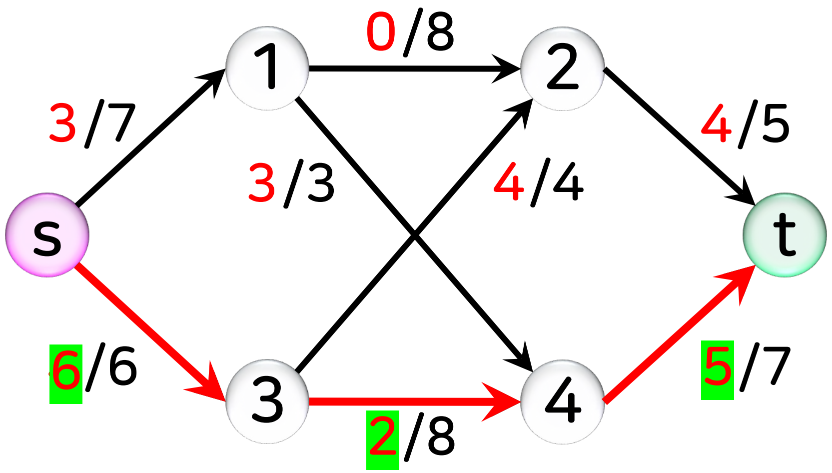
포드-풀커슨 알고리즘_예_1

증가 경로 s-1-4-t → 증가 경로의 여유량 = <1, 4>의 잔여 용량 = 3



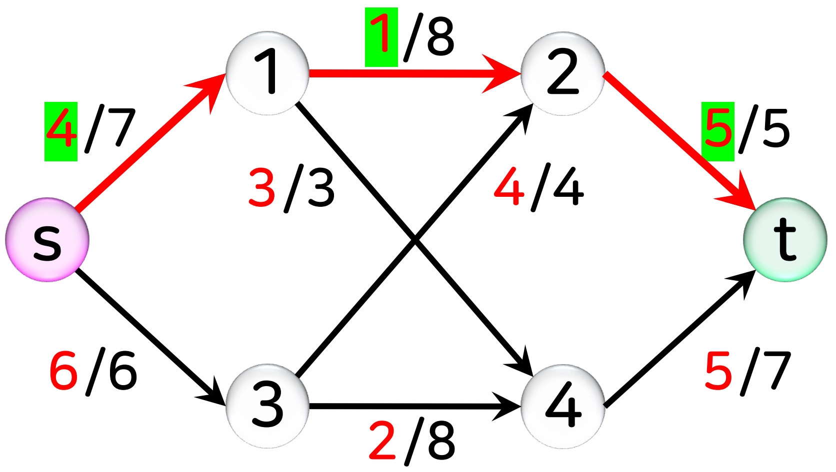
포드-풀커슨 알고리즘_예_1

증가 경로 s-3-4-t → 증가 경로의 여유량 = <s, 3>의 잔여 용량 = 2



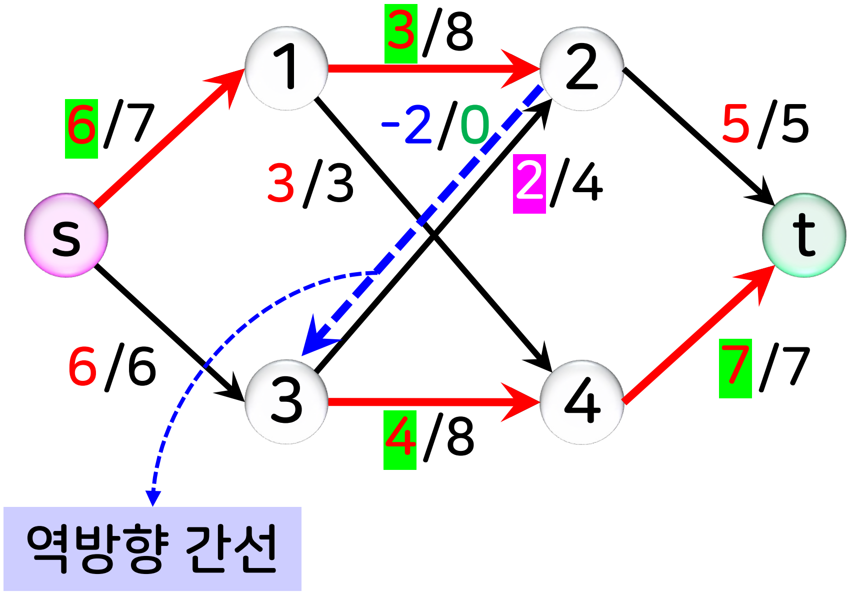
포드-풀커슨 알고리즘_예_1

증가 경로 s-1-2-t → 증가 경로의 여유량 = <2 ,t>의 잔여 용량 = 1



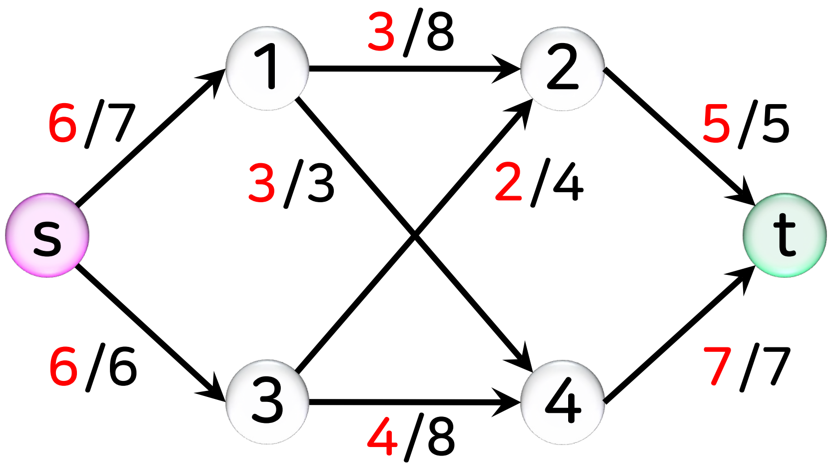
포드-풀커슨 알고리즘_예_1

증가 경로 s-1-2-3-4-t → 증가 경로의 여유량 = <4 ,t>의 잔여 용량 = 2



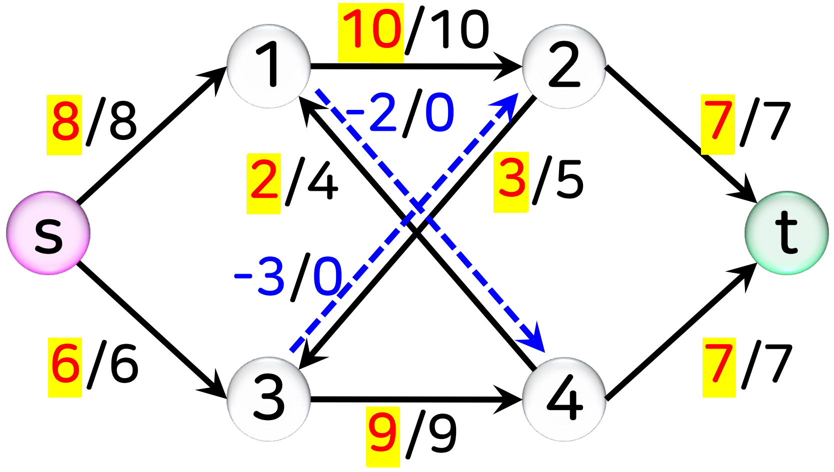
포드-풀커슨 알고리즘_예_1

더 이상의 증가 경로가 존재하지 않음 → 종료



최대 플로 F $\sum_{v \in V} f(s,v) = \sum_{w \in V} f(w,t)$ → 12

포드-풀커슨 알고리즘_예_2



s-1-2-3-4-t <2,3>의 잔여 용량: 5

s-3-4-1-2-t <3,4> <4,1>의 잔여 용량: 4

s-1-2-t <1,2>의 잔여 용량: 1

s-1-4-t <s,1> <4,t> 의 잔여 용량: 2

s-3-2-t <s,3> <2,t> 의 잔여 용량: 2

역방향 간선

최대 플로 F = 14

성능과 특징


```
Ford_Fulkerson ( N )
{
  모든 간선의 플로 f를 0으로 초기화;
  while ( 증가 경로 p가 존재 ) {
    Δ = min{ r(u, v) | p에 존재하는 <u, v> }
    for ( 증가 경로 p상의 모든 간선 <u, v> )
      if ( <u, v>가 순방향 간선 ) f(u, v) += Δ;
      else f(u, v) -= Δ;
  }
  F = ∑w∈V f(w,t);
  return (F);
}
```

최대 플로 값 F에 도달하기 위해
최소 1씩 증가한다면 최대 F번 반복
(용량이 정수인 경우)

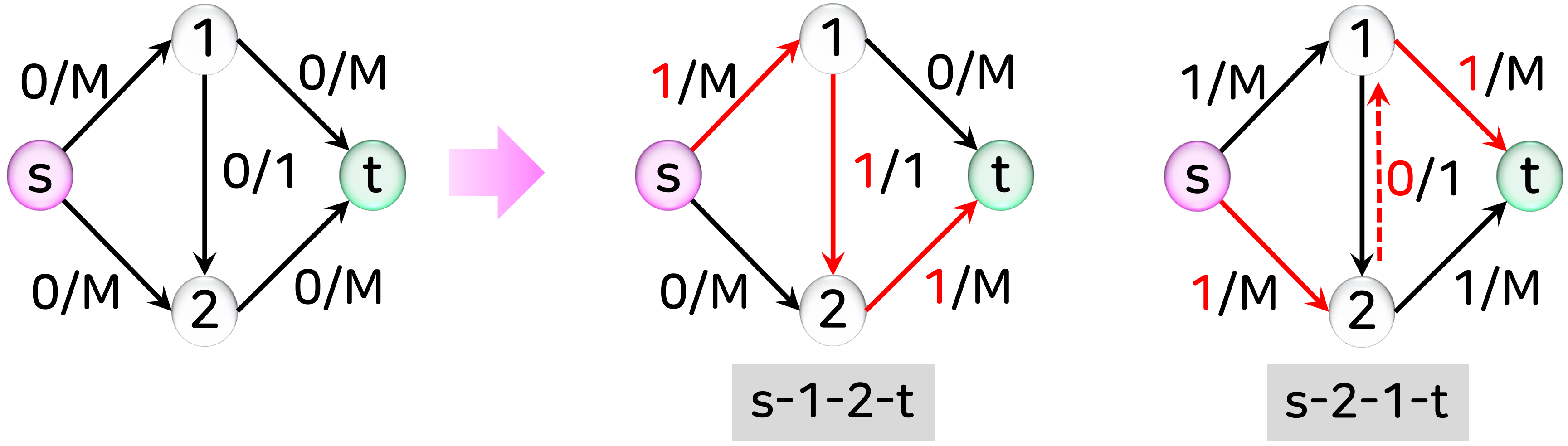
DFS/BFS 사용 → O(|E|)

O(|E|F)

성능과 특징

문제점

- 용량으로 무리수를 사용 → 알고리즘의 종료가 보장되지 못함
- 용량 M이 매우 큰 값이면 비효율적



✓ 증가 경로 s-1-2-t와 s-2-1-t를 각각 M번 찾음 → F=2M

성능과 특징

증가 경로의 선택은 DFS 또는 BFS 적용

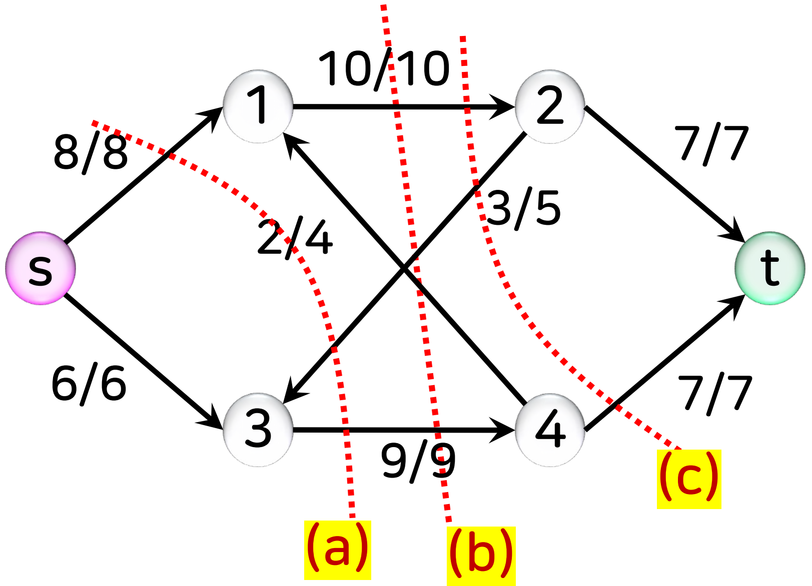
- 포드-풀커슨 알고리즘 → 기본적으로 DFS를 적용하여 증가 경로 선택
  - ✓ 에드몬즈-카프 알고리즘 → BFS 적용, O(|V||E|2), 포드-풀커슨의 문제 해결

커트 cut

$$S \subseteq V, s \in S, t \notin S, T = V - S$$

커트 → $\boxed{(S;T)} \cup (T;S)$

S의 정점에서 T의 정점으로 향하는 간선의 집합



1. 벨만-포드 알고리즘

- 음의 가중치를 갖는 간선이 존재해도 적용할 수 있는 단일 출발점 최단 경로 알고리즘
- O(|V||E|)

2. 플로이드 알고리즘

- 모든 정점 쌍 간의 최단 경로, 동적 프로그래밍 방법
- 경유할 수 있는 중간 정점의 범위를 하나씩 늘려가는 방식
- O(|V|3), 최단 경로 자체를 구하는 경우에는 P[i][j]에 경유하는 중간 정점을 저장

3. 포드-풀커슨 알고리즘

- 네트워크 플로 문제 → 네트워크에 대해서 소스에서 싱크로 보낼 수 있는 최대 플로 값을 구하는 문제
- 포드-풀커슨 방법 → 증가 경로가 더 이상 존재하지 않을 때가 반복적으로 찾아서 플로를 증가시키는 방법, O(|E|F)
- 순방향 간선, 역방향 간선, 잔여 용량, 증가 경로의 여유량, 커트

ALGORITHM 알고리즘

다음시간에는

Lecture 11

동적 프로그래밍

컴퓨터과학과 | **이관용 교수**