알고리즘(Algorithms)

-그래프 알고리즘 2-

동아대학교 소프트웨어대학 컴퓨터공학과

2025년 2학기

임 한 신

공지 사항

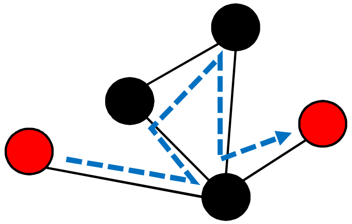
- [**NVIDIA RTX AI Campus Seminar**](**11월 24일 수업 대체 예정**)
- 가. 일시: **2025.11.24.(월) 14:00~16:00**
- 나. 장소: 승학캠퍼스 청춘홀(메인홀), 경동홀(이원중계홀)
- 다. 대상: 동아대학교 재학생
- 라. 내용
- - NVIDIA GPU 와 AI 트렌드 / 엔비디아 이득우 상무
- - RTX AI 데모 / 엔비디아 최익태 상무
- - 현업 개발자 세션 / TBD
- - 폐회사 및 럭키드로우 / 엔비디아 김승규 대표 외

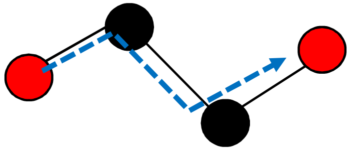
- [**부대행사 AI 체험부스 운영 - NVIDIA / HP**]
- - 장소: 공대 5호관 로비 및 S06-607 Complex Hall
- - 일시: 2025.11.24.(월) 11:00~15:00

** Nvidia RTX AI Campus 세미나가 수도권 중심으로 진행되다가, 부산에서 처음으로 동아대학교에서 개최됩니다.

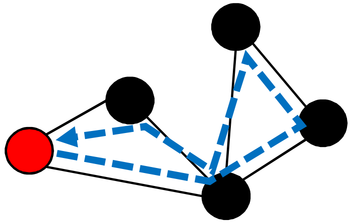
그래프 알고리즘(Graph Algorithms)

- 경로
  - 이동할 수 있는 연결된 정점들의 나열

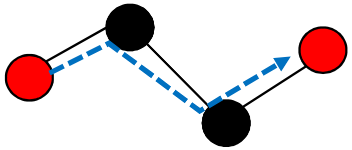




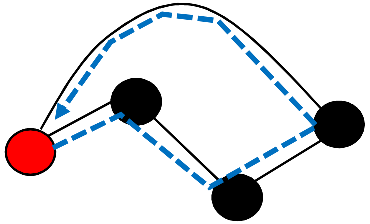
- 사이클
  - 시작 정점으로 돌아가는 경로



- 단순 경로
  - 사이클을 포함하지 않는 경로

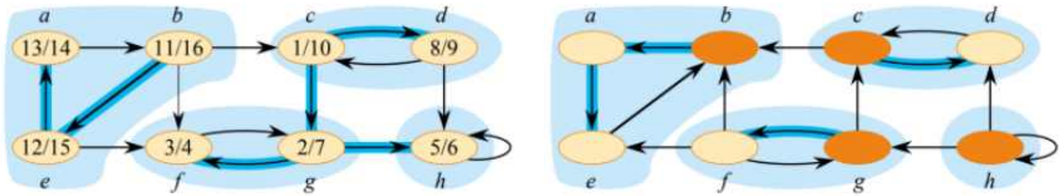


- 단순 사이클
  - 사이클을 포함하지 않는 사이클

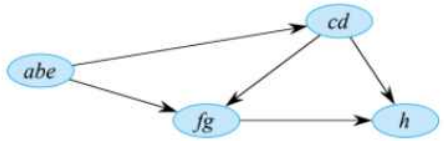


그래프 알고리즘(Graph Algorithm)

- 강연결 요소
  - 방향 그래프 $G=(V, E)$에서 V의 모든 정점 쌍 (u, v)에 대해 경로 $u \leadsto v$와 경로 $v \leadsto u$가 존재하면 그래프 G는 강하게 연결되어 있다고 함
  - 즉, 어떤 두 정점을 잡더라도 양방향으로 서로에게 이르는 경로가 존재하면 강하게 연결되었다고 함
  - 방향 그래프에서 강하게 연결된 부분 그래프를 강연결 요소(Strongly connected component)라고 함



(a) (b)

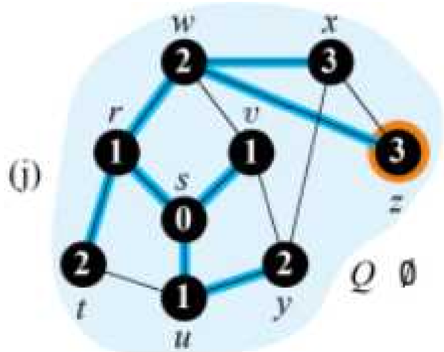


(c)

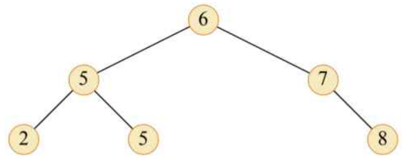
*G*의 각각의 강연결 요소에서 모든 간선을 수축시켜서 얻어진 비순환 요소 그래프

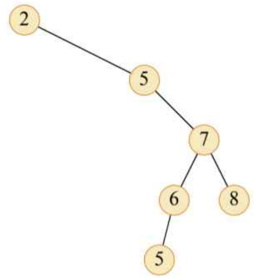
그래프 알고리즘(Graph Algorithms)

- 신장 트리(Spanning Tree)
  - 무방향 그래프 $G=(V, E)$의 신장 트리는 정점 집합 V를 그대로 두고 간선을 $|V|-1$개만 남겨 트리가 되도록 만든 것
  - 모든 정점을 포함하면서 사이클이 없는 최소 연결 부분 그래프
  - 무방향 그래프의 너비 우선 트리, 깊이 우선 트리도 신장 트리의 한 종류임



너비 우선 트리

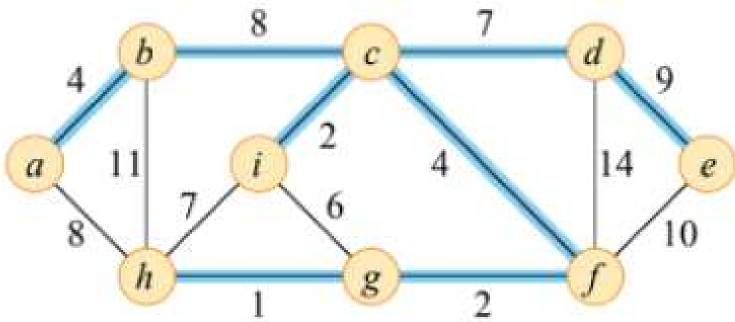




※ 트리는 항상 정점보다 간선이 하나 적은 연결 그래프이다.

그래프 알고리즘(Graph Algorithms)

- 최소 신장 트리(Minimum Spanning Tree)
  - 간선들이 가중치를 갖는 무방향 그래프에서 간선 가중치의 합이 가장 작은 트리를 최소 신장 트리(MST)라 함

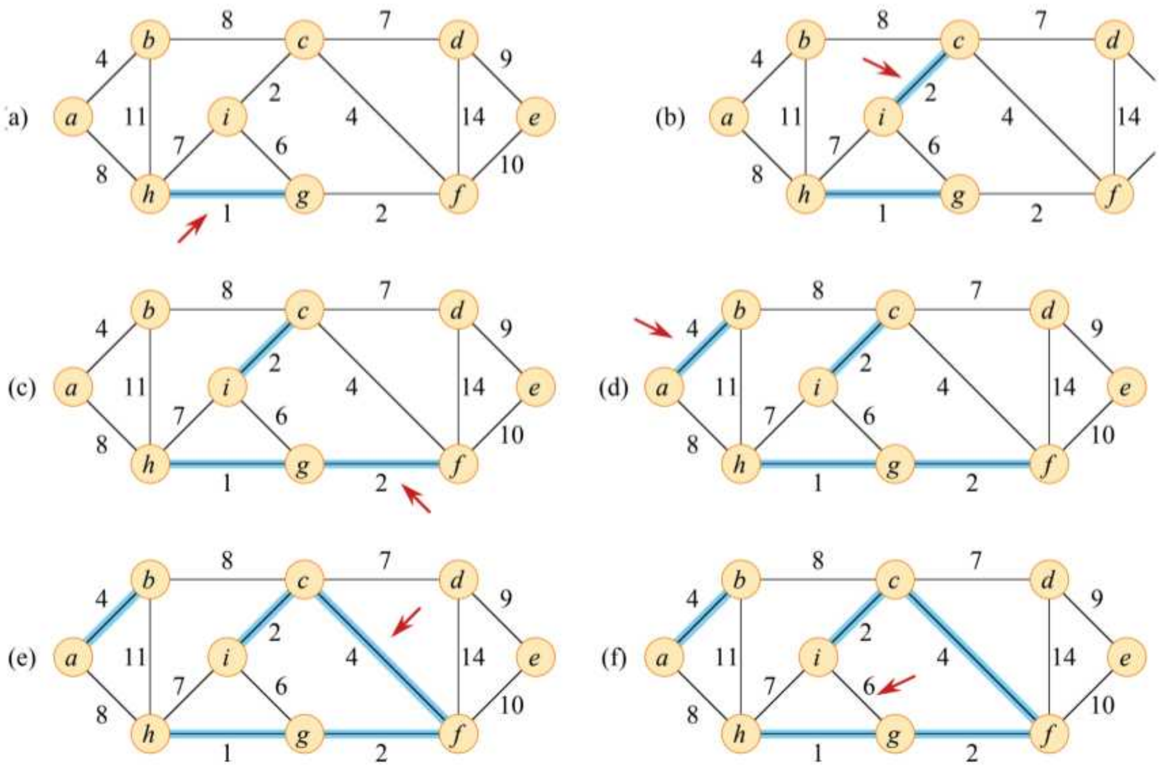


전체 가중치가 37인 최소 신장 트리의 예

  - 최소 신장 트리를 구하는 대표적인 알고리즘에는 그리디 알고리즘의 일종인 크루스칼 알고리즘(Kruskal's algorithm)과 프림 알고리즘(Prim’s Algorithm)이 있음

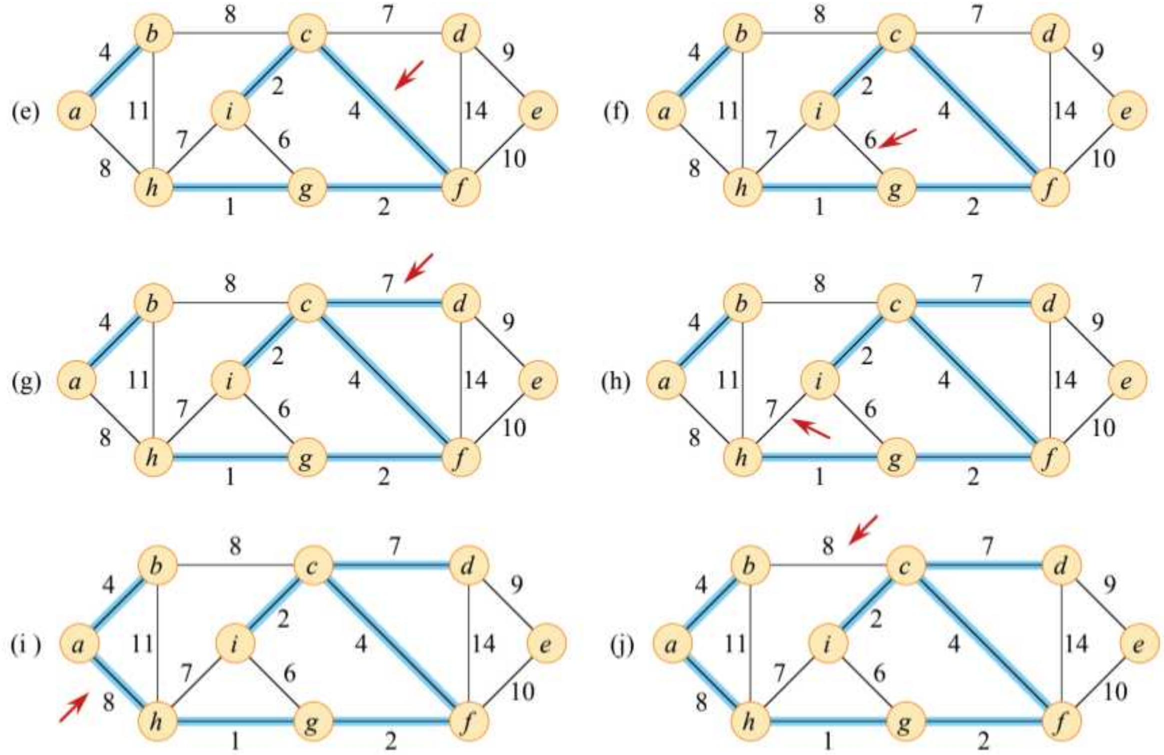
그래프 알고리즘(Graph Algorithms)

- 최소 신장 트리(Minimum Spanning Tree)
  - 크루스칼 알고리즘(Kruskal's algorithm)



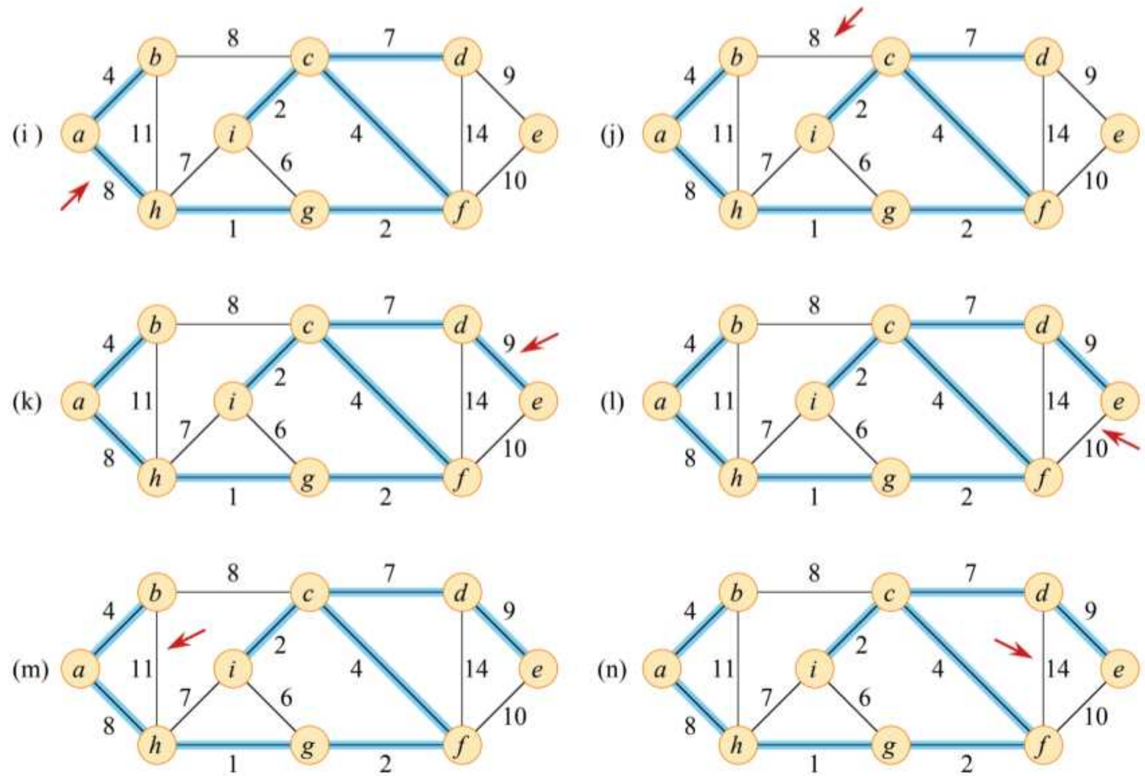
그래프 알고리즘(Graph Algorithms)

- 최소 신장 트리(Minimum Spanning Tree)
  - 크루스칼 알고리즘(Kruskal's algorithm)



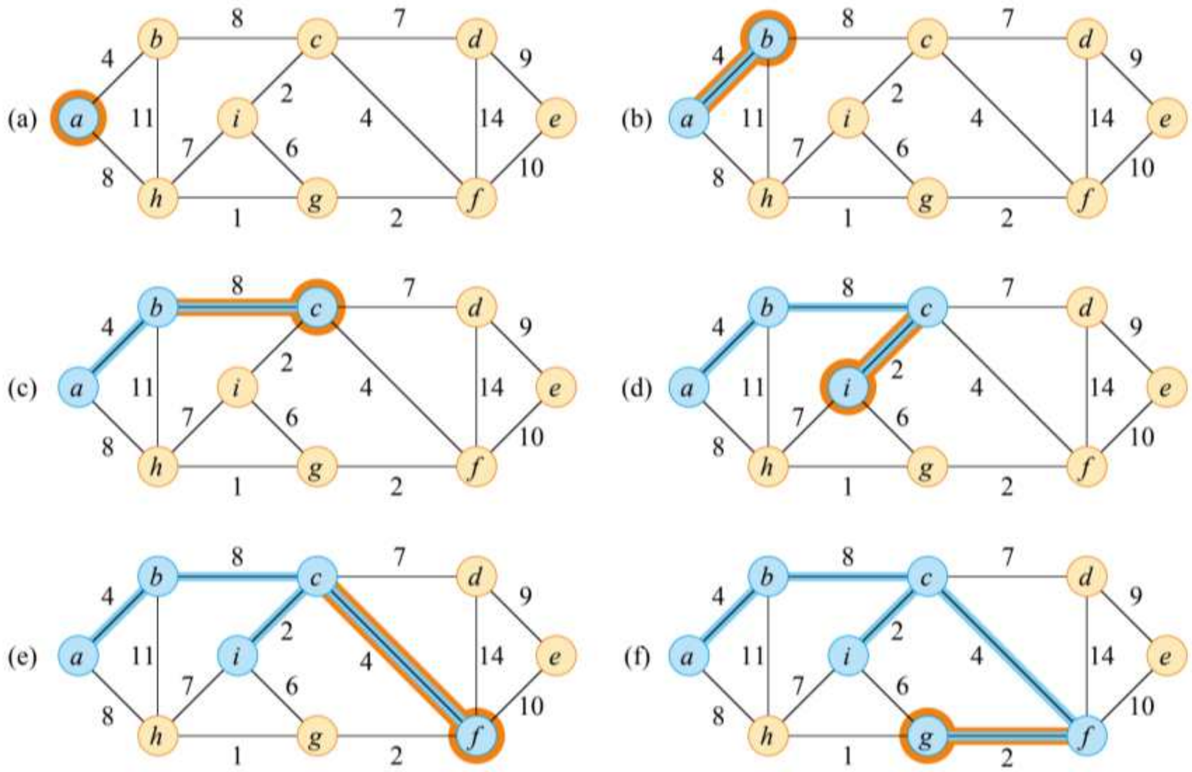
그래프 알고리즘(Graph Algorithms)

- 최소 신장 트리(Minimum Spanning Tree)
  - 크루스칼 알고리즘(Kruskal's algorithm)



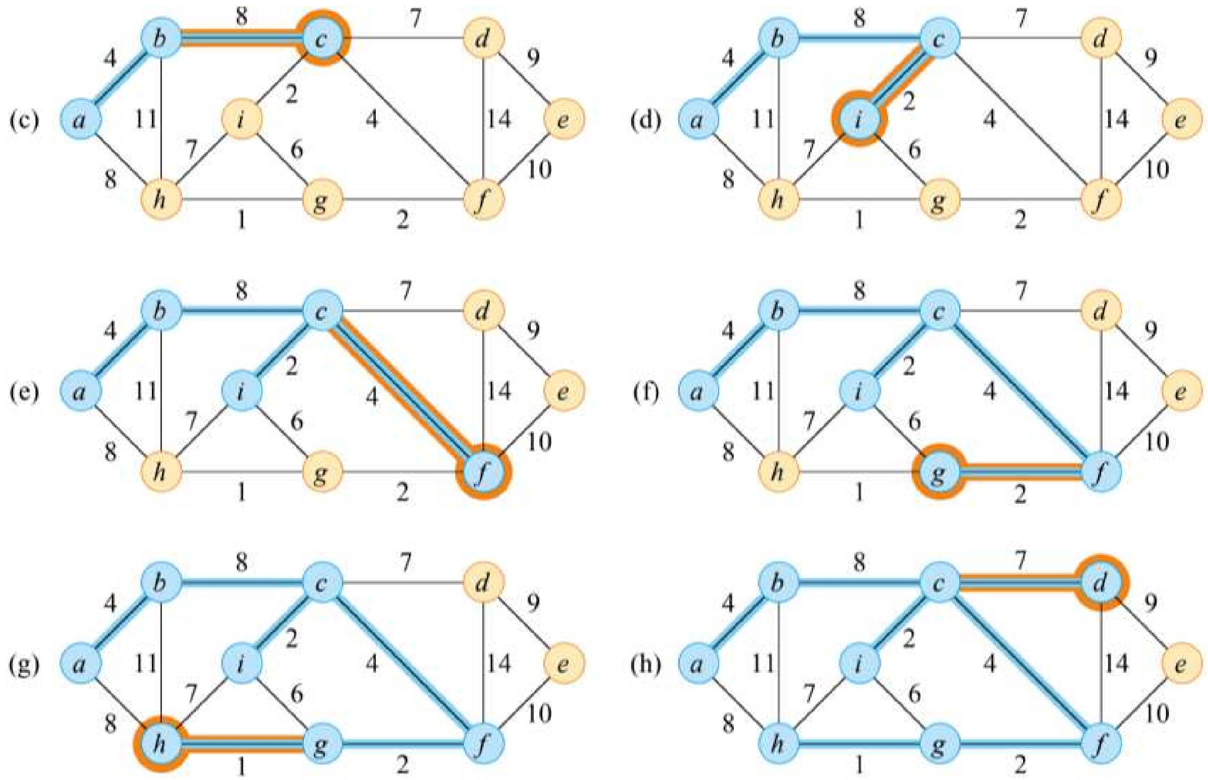
그래프 알고리즘(Graph Algorithms)

- 최소 신장 트리(Minimum Spanning Tree)
  - 프림 알고리즘(Prim's Algorithm)



그래프 알고리즘(Graph Algorithms)

- 최소 신장 트리(Minimum Spanning Tree)
  - 프림 알고리즘(Prim's Algorithm)



수행 시간 : $O(E\log V)$

단일 출발점 최단 경로

- 최단 경로
  - 가중 방향 그래프 $G = (V, E)$와 각 간선에 실수 가중치를 부여하는 가중 함수 $w : E \rightarrow R$이 주어졌을 때, 경로 $p = <v_0, v_1,\ldots, v_k>$의 가중치(weight) $w(p)$는 그 경로를 이루는 간선들의 가중치의 합임

$$w(p) = \sum_{i=0}^{k} w(v_{i-1}, vi)$$

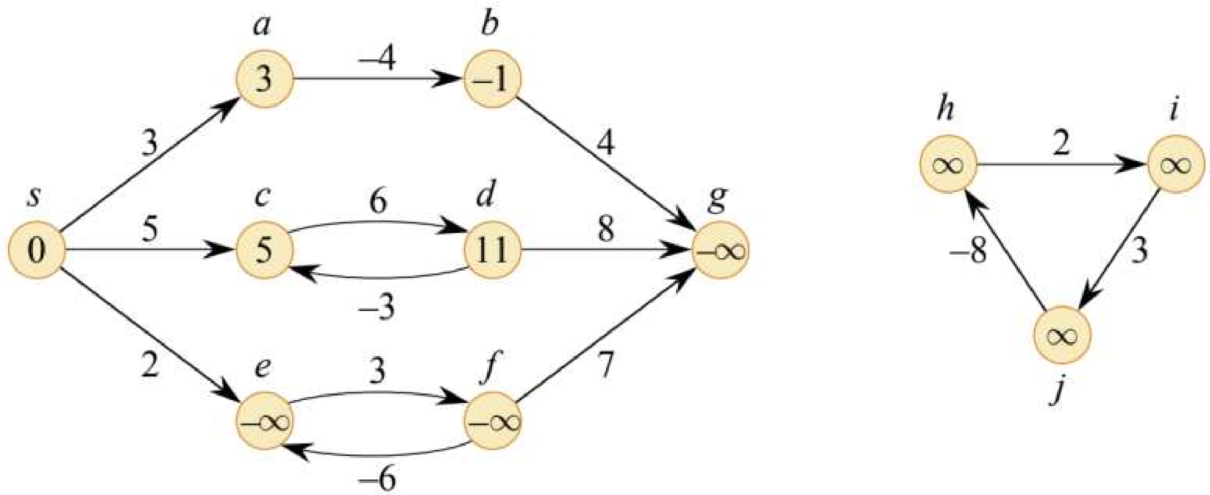
  - u에서 v까지의 모든 가능한 경로 중 경로의 가중치가 가장 작은(최단 경로 가중치 (shortest-path weight)) 경로를 최단 경로라고 함
  - 간선의 가중치는 거리가 아닌 시간, 비용, 벌금, 손실 또는 경로를 따라 선형적으로 축적되고 최소화하고자 하는 양을 나타낼 수 있음
  - 너비 우선 탐색 알고리즘은 각 간선이 단위 가중치(1)를 갖는 그래프에서 최단 경로를 찾는 알고리즘임

단일 출발점 최단 경로

- 단일 출발점 최단 경로 문제
  - 그래프 $G = (V, E)$의 출발 정점 $s \in V$에서 모든 정점 $v \in V$로의 최단 경로를 찾는 문제를 단일 출발점 최단 경로 문제라고 함
  - 최단 경로의 부분 경로들은 최단 경로들임(최적 부분 구조 특성을 가짐)
  - 다익스트라(Dijkstra) 알고리즘은 도로망에서와 같이 모든 간선의 가중치가 음이 아닌 경우에 단일 출발점 최단 경로 문제를 해결
  - 벨만-포드(Bellman-Ford) 알고리즘은 간선이 음의 가중치를 가질 수 있는 경우의 단일 출발점 최단 경로 문제를 해결

단일 출발점 최단 경로

- 단일 출발점 최단 경로 문제
  - 순환
    - 최단 경로들은 순환을 포함하지 않는 단순 경로라고 가정함
    - 순환이 없는 어떤 경로든지 최대 $|V|$개의 서로 다른 정점을 포함하고 있으므로 최대 $|V|$-1개의 간선을 포함



방향 그래프에서 음의 간선 가중치와 순환을 포함한 경우의 예시

단일 출발점 최단 경로

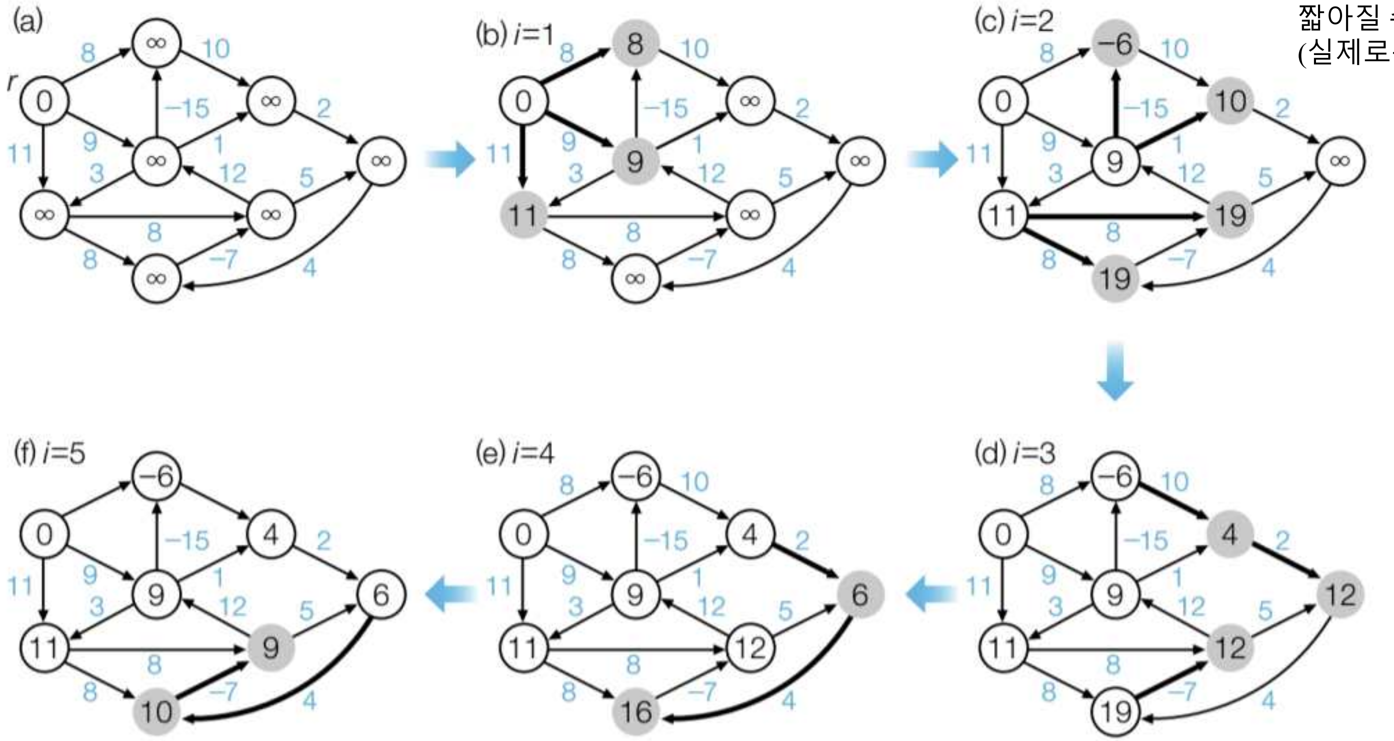
- 벨만 포드 알고리즘(Bellman-Ford algorithm)
  - 간선의 가중치가 음수인 일반적인 경우에 단일 출발점 최단 경로 문제를 해결
  - 벨만-포드 알고리즘은 간선을 최대 1개 사용하는 최단 경로, 간선을 최대 2개 사용하는 최단 경로,…, 간선을 최대 n-1개 사용하는 최단 경로까지 구해나감

단일 출발점 최단 경로

- 벨만 포드 알고리즘(Bellman-Ford algorithm)
  - 벨만-포드 알고리즘의 작동 예

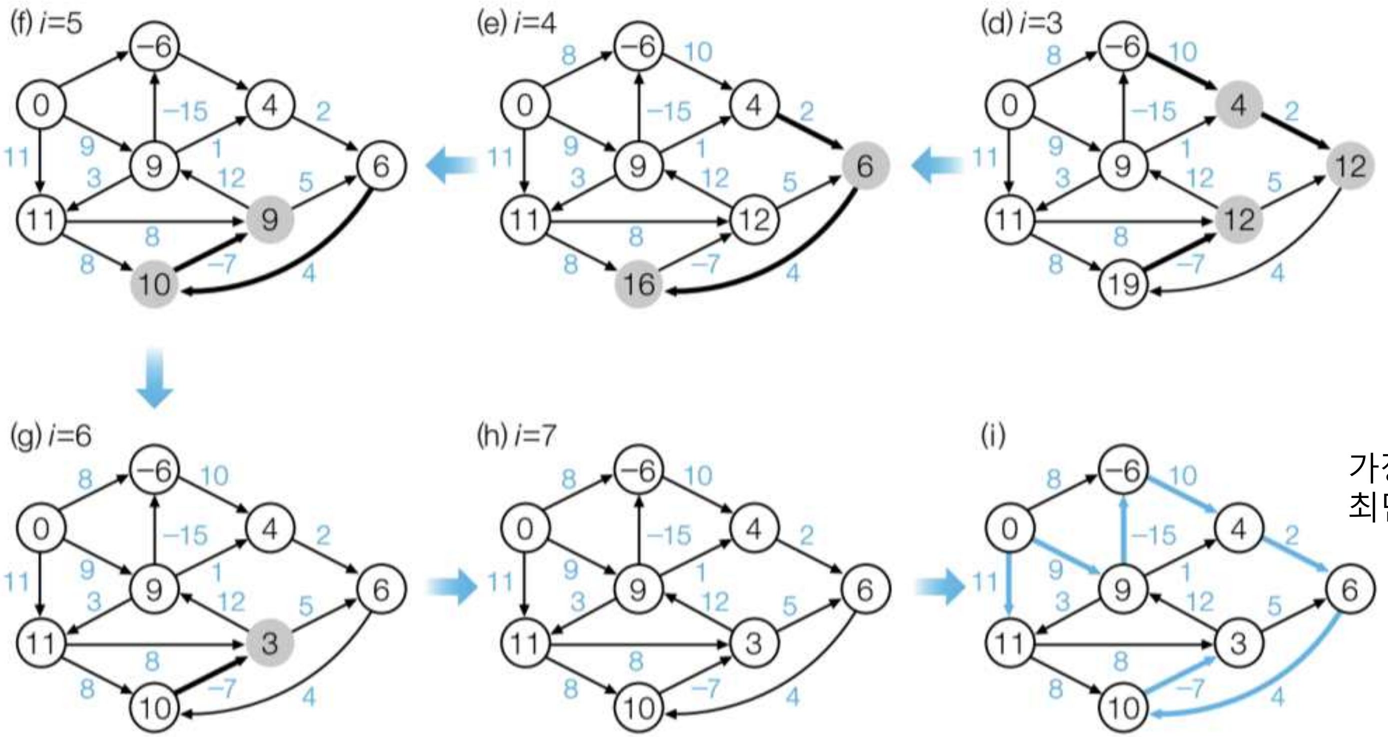
모든 간선을 한 번씩 살피면서 해당 간선으로 인해 앞에서 설정한 최단 거리가 더 짧아질 수 있는지 확인
(실제로는 변동이 생긴 정점의 간선들만)

시작 정점 r만 최단 거리0으로 초기화하고 다른 정점의 최단 거리는 모두 ∞ 로 초기화



단일 출발점 최단 경로

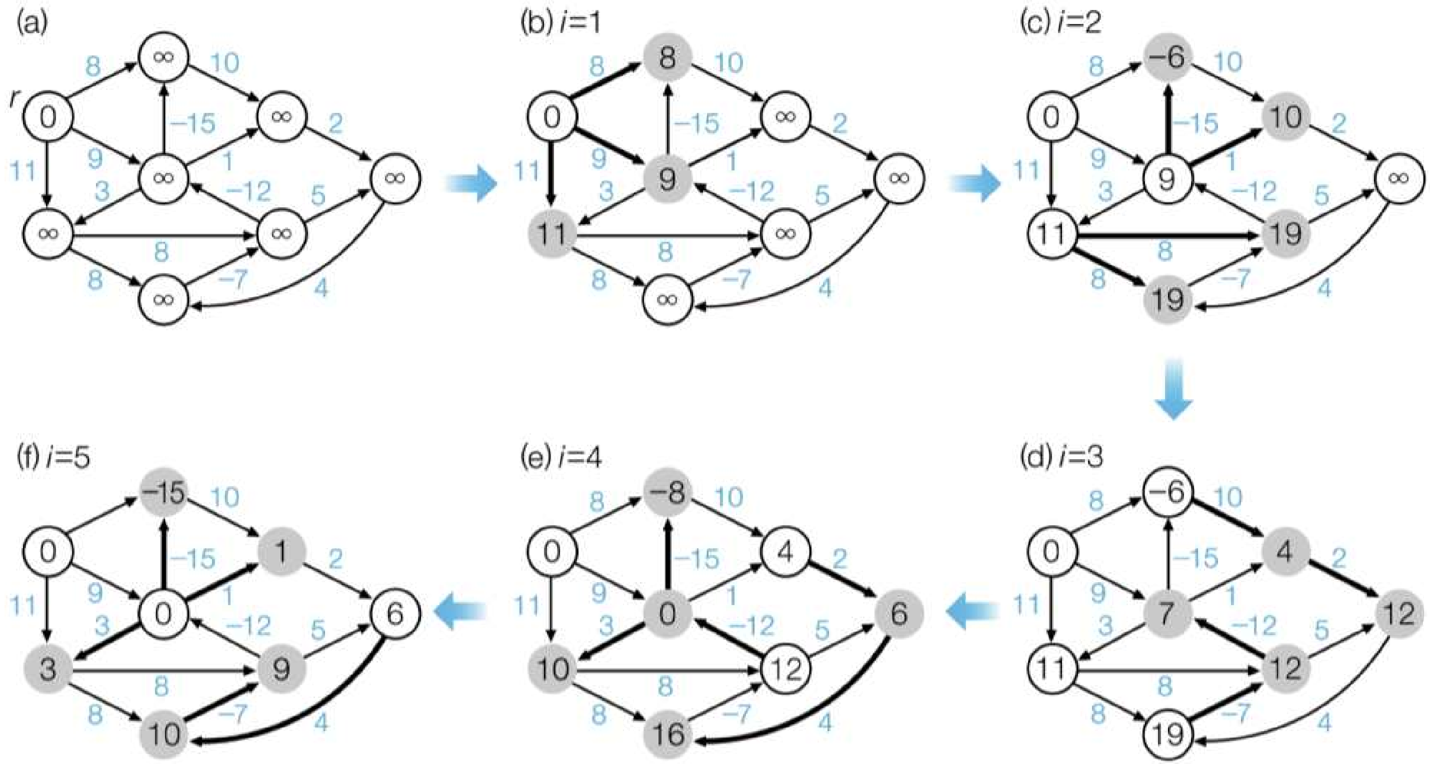
- 벨만 포드 알고리즘(Bellman-Ford algorithm)
  - 벨만-포드 알고리즘의 작동 예



가장 많은 수의 간선을 사용하는
최단 경로는 6개의 간선 사용

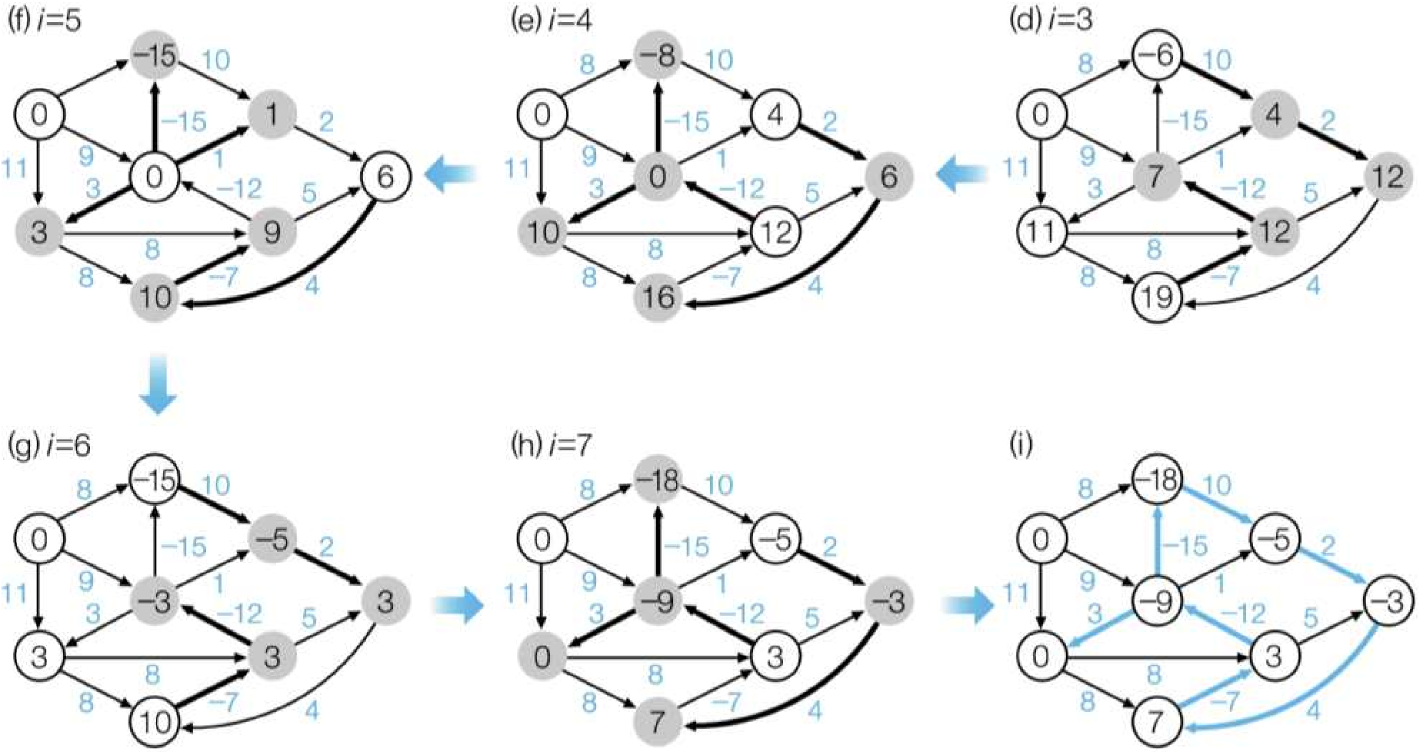
단일 출발점 최단 경로

- 벨만 포드 알고리즘(Bellman-Ford algorithm)
  - 음의 순환을 갖는 경우 벨만-포드 알고리즘의 작동 예



단일 출발점 최단 경로

- 벨만 포드 알고리즘(Bellman-Ford algorithm)
  - 음의 순환을 갖는 경우 벨만-포드 알고리즘의 작동 예



단일 출발점 최단 경로

- 벨만 포드 알고리즘(Bellman-Ford algorithm)
  - 수행 시간 분석

```
BellmanFord(G, r):
    for each u ∈ V
        u.dist ← ∞
    r.dist ← 0
  ❶ for i ← 1 to |V|−1
      ❷ for each (u, v) ∈ E
            if (u.dist + w_uv < v.dist)
              ❸ v.dist ← u.dist + w_uv
                 v.prev ← u
    ▷ 음의 사이클 존재 여부 확인
  ❹ for each (u, v) ∈ E
        if (u.dist + w_uv < v.dist) output "음의 사이클 발견! 해 없음"
```

$\Theta(VE)$

$\Theta(E)$

벨만 포드 알고리즘의 수행시간 : $\Theta(VE)$

단일 출발점 최단 경로

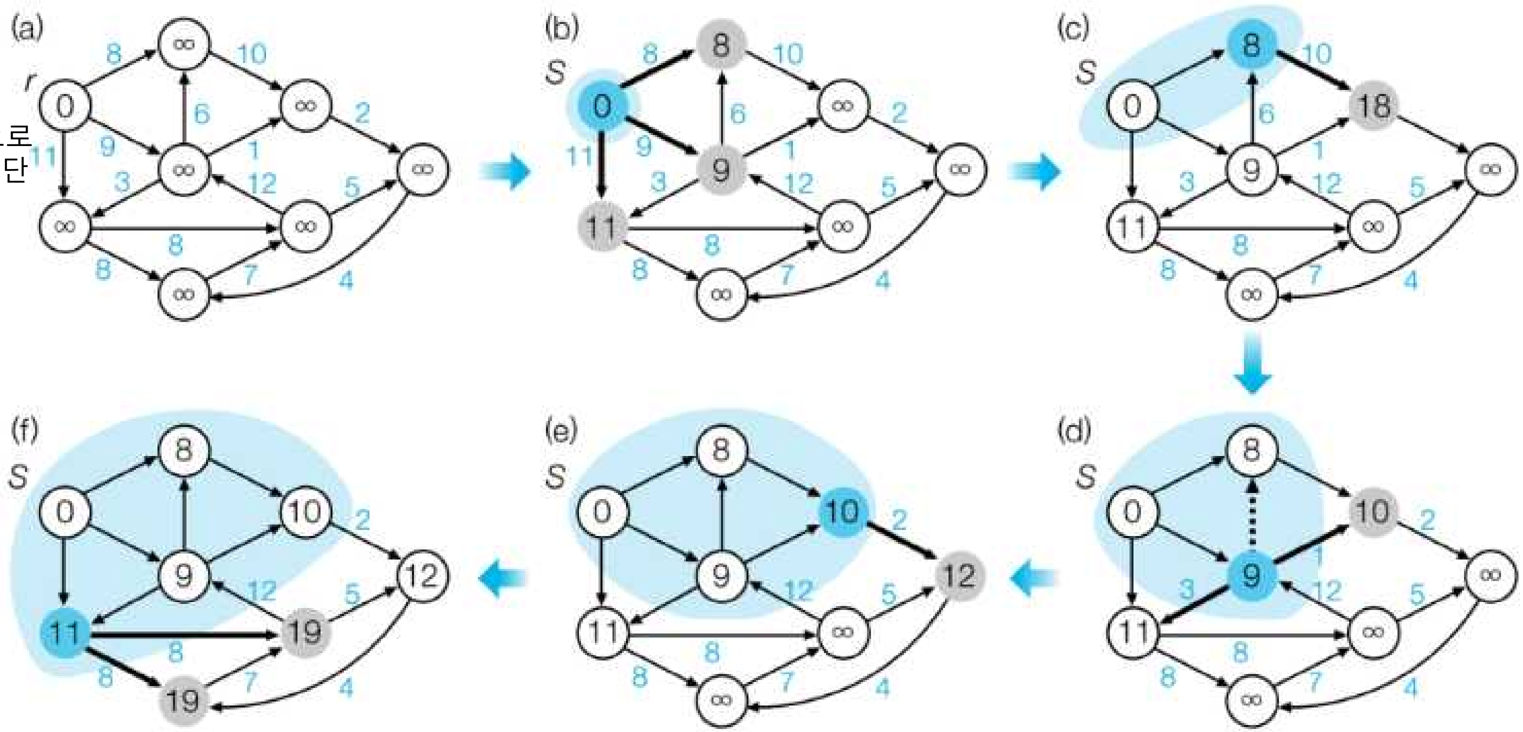
- 다익스트라 알고리즘(Dijkstra algorithm)
  - 입력 그래프 G=(V, E)에서 간선들의 가중치가 모두 0이상인 경우의 최단 경로 알고리즘
  - 최소 신장 트리(Minimum Spanning Tree)를 구하는 프림 알고리즘과 원리가 거의 같음
  - 프림 알고리즘과의 차이점은
    - 프림 알고리즘은 정점 v를 신장 트리(spanning tree)에 연결하는 최소 비용을 저장
    - 다익스트라 알고리즘은 정점 r에서 정점 v에 이르는 최단 거리를 저장

  - 단일 출발점 최단 경로에서
    - 임의의 두 정점 간에 경로가 존재하지 않으면 이 경로의 길이는 ∞로 간조
    - 임의의 두 정점 사이에 간선이 없으면 가중치가 ∞인 간선이 있다고 간주

단일 출발점 최단 경로

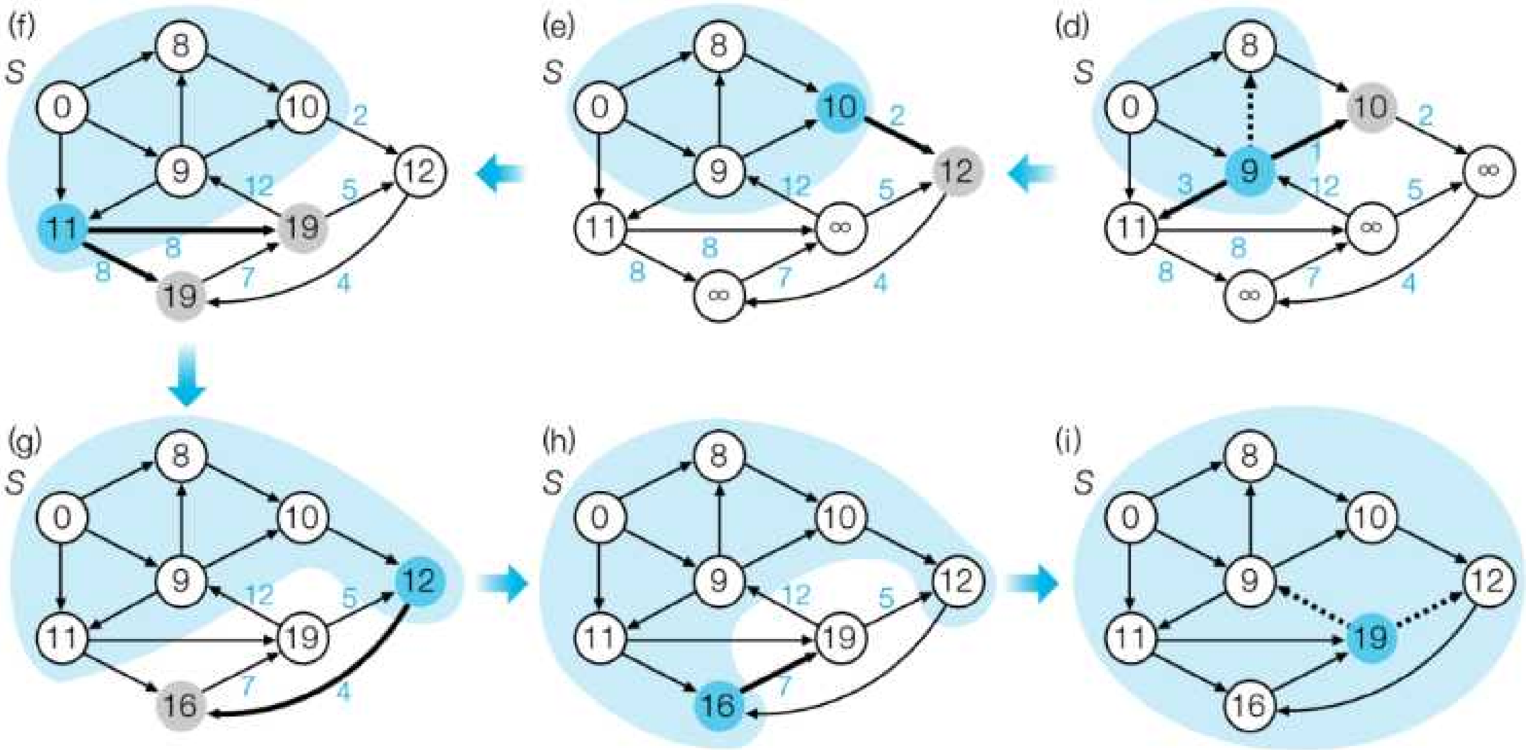
- 다익스트라 알고리즘(Dijkstra algorithm)
  - 다익스트라 알고리즘의 작동 예

시작 정점 r만 최단 거리0으로 초기화하고 다른 정점의 최단 거리는 모두 ∞ 로 초기화



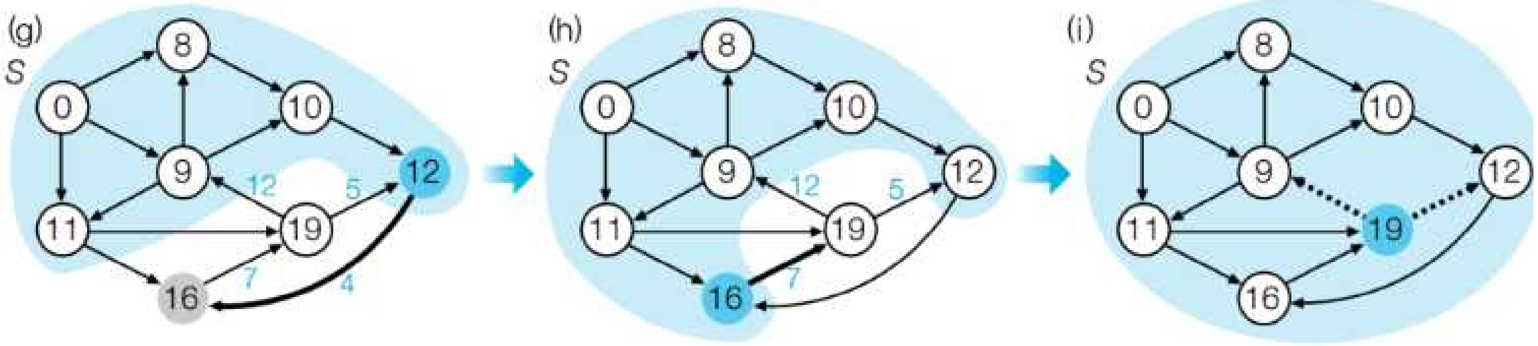
단일 출발점 최단 경로

- 다익스트라 알고리즘(Dijkstra algorithm)
  - 다익스트라 알고리즘의 작동 예

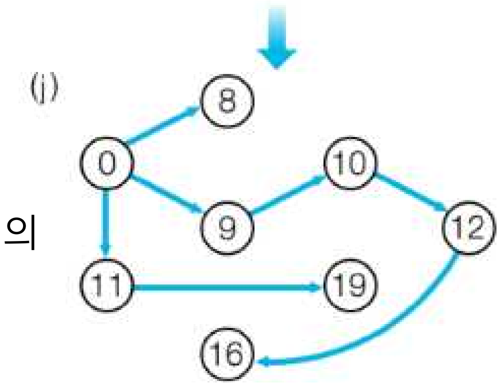


단일 출발점 최단 경로

- 다익스트라 알고리즘(Dijkstra algorithm)
  - 다익스트라 알고리즘의 작동 예

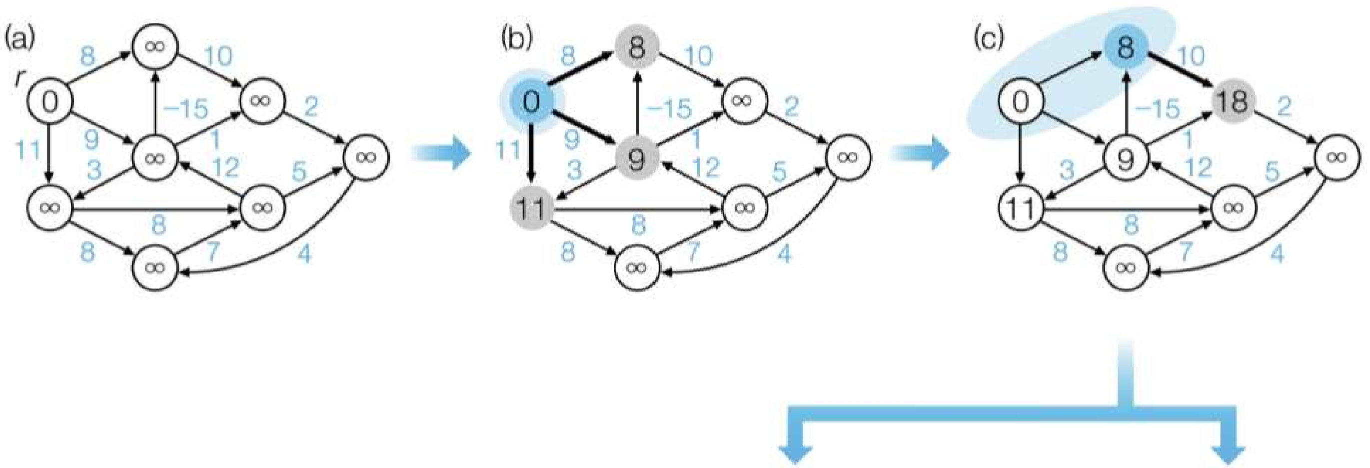


최단 경로 트리 :
시작 정점에서 모든 정점까지의
최단 경로를 나타내는 트리

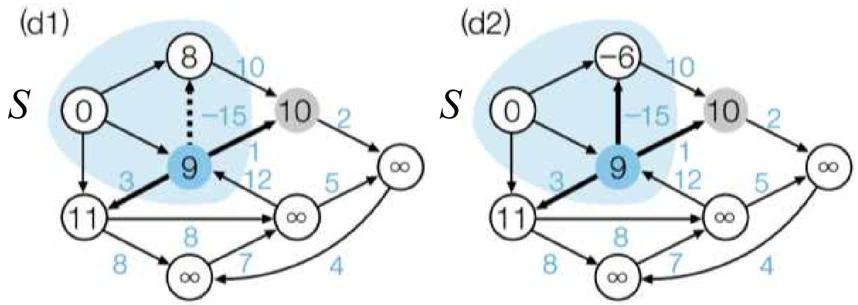


단일 출발점 최단 경로

- 다익스트라 알고리즘(Dijkstra algorithm)
  - 다익스트라 알고리즘이 제대로 작동하지 않는 예



다익스트라 알고리즘은
집합 S에 한 번 포함된 정점 및 연결된 정점의
최단 거리를 다시 계산하지 않음



단일 출발점 최단 경로

- 다익스트라 알고리즘(Dijkstra algorithm)
  - 수행 시간 분석

```
Dijkstra(G, r):
▷ G=(V, E): 주어진 그래프
▷ r: 시작으로 삼을 정점
    S ← ø                          ▷ S: 정점 집합
    for each u ∈ V
        u.dist ← ∞
    r.dist ← 0
    while (S≠V)                    ▷ n회 순환
        u ← extractMin(V−S)
        S ← S∪{u}
        for each u.adjlist         ▷ u.adjlist: 정점 u로부터 연결된 정점들의 집합
            if (v ∈ V−S and u.dist+w_uv < v.dist)
                ❶ v.dist ← u.dist+w_uv
                  v.prev ← u

extractMin(Q):
    집합 Q에서 u.dist 값이 가장 작은 정점 u를 리턴한다.
```

최소 신장 트리(MST)를 구하는 프림 알고리즘과 원리가 거의 같고 수행 시간도 동일하게 $O(E\log V)$ 임

단일 출발점 최단 경로

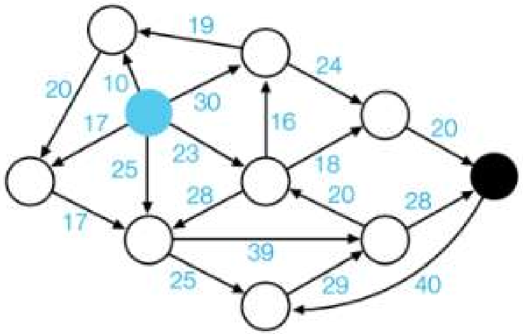
- A* 알고리즘(A-star algorithm)
  - 현재까지의 비용 $g(x)$에 미래 비용의 추정 $h(x)$을 더해 가장 유망한 경로를 먼저 탐색하는 휴리스틱 기반 최단 경로 알고리즘
  - 즉, 아래의 $f(x)$를 계산해서 값이 가장 작은 노드부터 우선 탐색

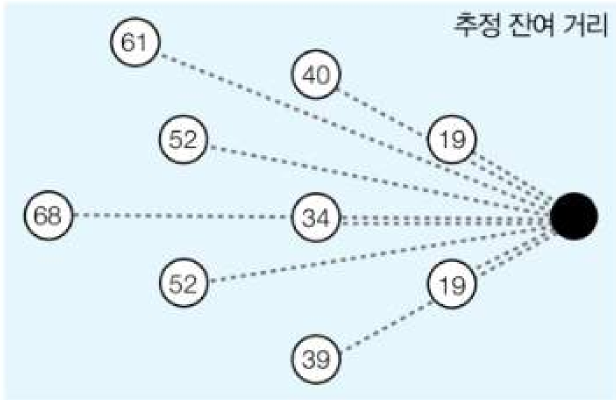
$$f(x) = g(x) + h(x)$$

    - $g(x)$: 시작점 → 현재 노드까지 실제 비용
    - $h(x)$: 현재 노드 → 목표까지 추정 비용(휴리스틱)
    - $f(x)$: A*가 평가하는 값. 작을수록 먼저 탐색

단일 출발점 최단 경로

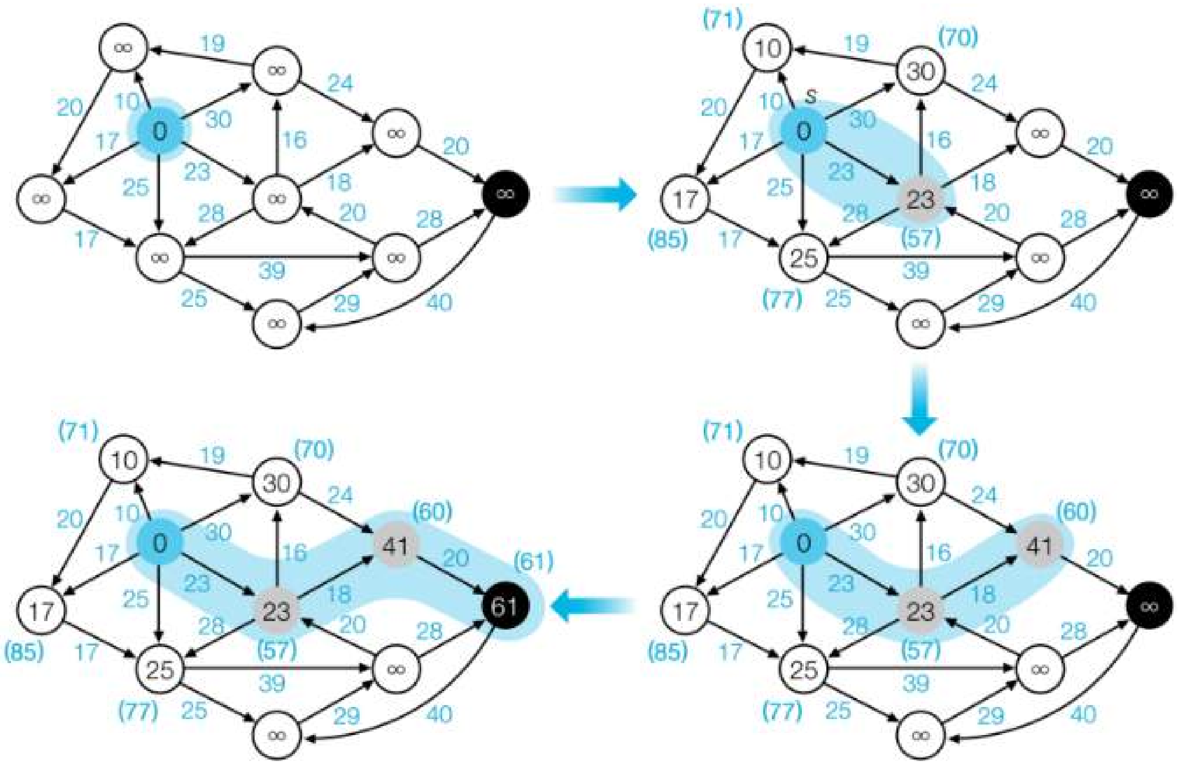
- A* 알고리즘(A-star algorithm)
  - A* 알고리즘을 이용한 최단 경로 찾기의 예





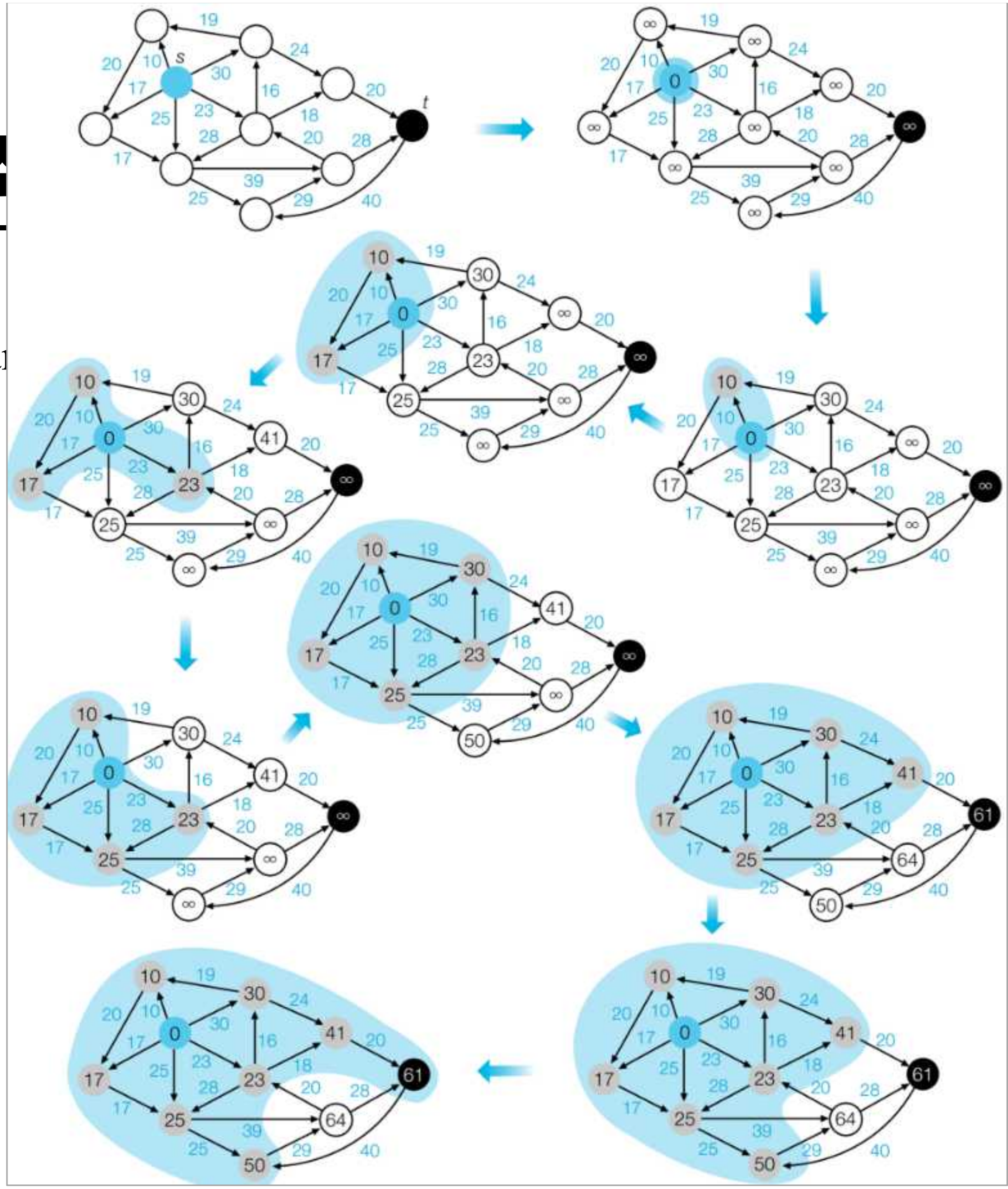
단일 출발점 최단 경로

- A* 알고리즘(A-star algorithm)
  - A* 알고리즘을 이용한 최단 경로 찾기의 예



단일 출발점 최

- A* 알고리즘(A-sta



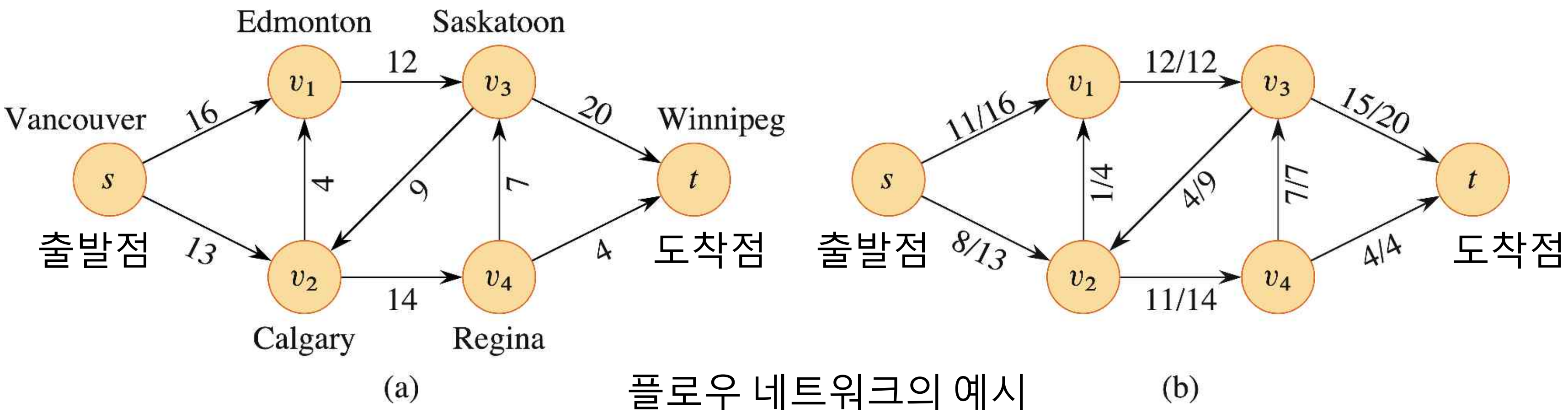
다익스트라 알고리즘을
이용한 최단 경로 찾기

단일 출발점 최단 경로

- A* 알고리즘(A-star algorithm)
  - A*가 최적해를 보장할 조건
    - 추정거리 $h(x)$는 정점 x에서 목적점에 이르는 실제 최단 잔여 거리보다는 크지 않아야 함
    - 예) x로부터 목적지까지의 직선거리는 실제 최단 잔여 거리보다는 크지 않음
  - 다익스트라 알고리즘과의 차이점
    - 다익스트라 알고리즘은 상황에 관계없이 $h(x) = 0$인 A* 알고리즘의 특수한 경우로 볼 수 있음
    - 다익스트라 알고리즘은 특정한 목적점 하나를 명시하지 않고 하나의 시작점만 주면 다른 모든 정점에 이르는 최단 거리를 구함

최대 플로우(Maximum Flow)

- 플로우 네트워크(Flow network)
  - 플로우 네트워크(flow network) G=(V, E)는 각 간선 (u, v) ∈ E 가 음이 아닌 용량(capacity) $c(u, v) \geq 0$을 갖는 방향 그래프
  - 출발점(source) s와 도착점(sink) t를 가짐
  - 하나의 정점에서 다른 정점으로 흐르는 플로우가 음수는 아니어야 하고 주어진 용량도 초과할 수 없음
  - 각 정점(vertex)에 들어오는 플로우와 나가는 플로우는 같음



플로우 네트워크의 예시

최대 플로우(Maximum Flow)

- 플로우 네트워크(Flow network)의 정의
  - $G = (V, E)$를 용량 함수 c를 가지는 플로우 네트워크라 하자. 그리고 s를 플로우 네트워크의 출발점, t를 도착점이라 하자. G의 플로우는 다음 두 가지 특성을 만족시키는 실수 함수 $f : V \text{x} V \rightarrow R$ 이다.
    - 용량 제약조건 : 모든 $u, v \in V$ 에 대해 다음을 만족시켜야 한다.

      $0 \le f(u, v) \le \mathrm{c}(u, v)$

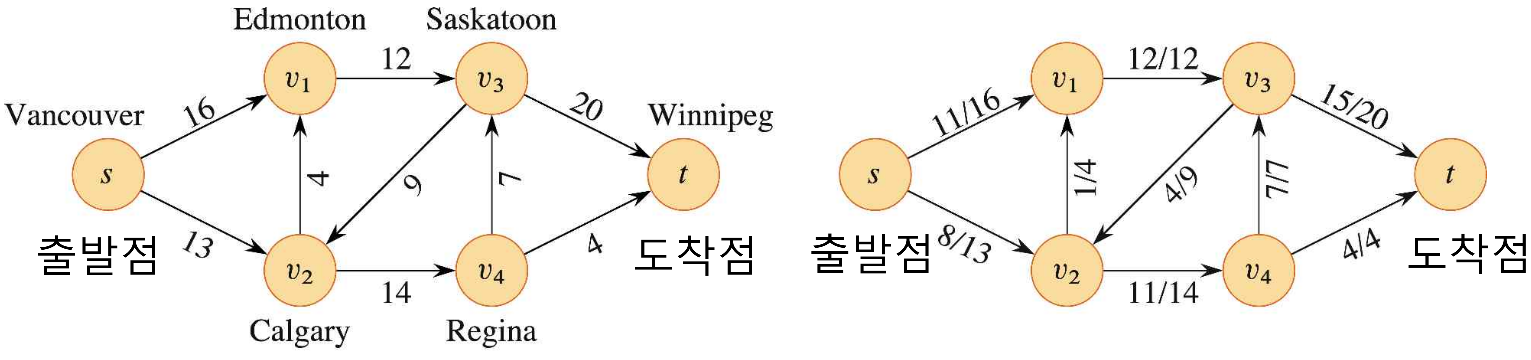
      하나의 정점에서 다른 정점으로 흐르는 플로우가 음수는 아니어야 하고 주어진 용량도 초과해서는 안된다.
    - 플로우 보존 : 모든 $u \in V - \{s, t\}$에 대해 다음을 만족시켜야 한다.

      $$\sum_{u \in V} f(v, u) = \sum_{u \in V} f(u, v)$$

      즉, 출발점이나 도착점이 아닌 하나의 정점으로 들어가는 플로우의 전체 양이 그 정점으로부터 나오는 플로우의 전체 양과 같아야 한다.

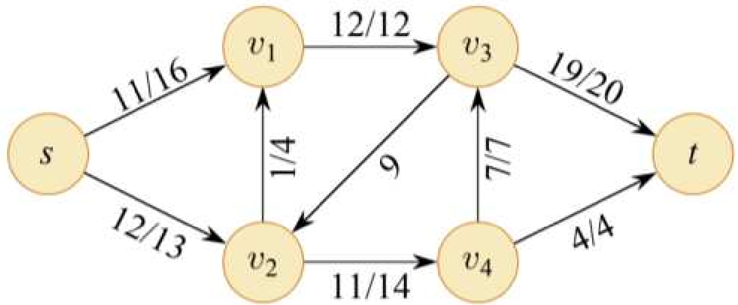
최대 플로우(Maximum Flow)

- 최대 플로우 문제(Maximum-flow problem)
  - 출발점 s와 도착점 t를 가지는 플로우 네트워크 G에서 최댓값을 가지는 플로우를 찾는 것



(a) (b)

- 응용 분야
  - 다양한 네트워크 문제
  - 자원 분배 문제
  - 영상 분할
  - 기타 등등



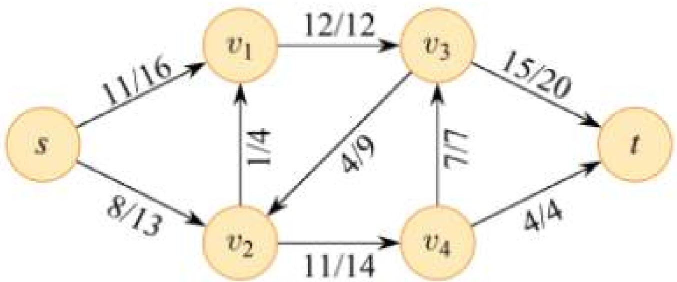
최대 플로우

최대 플로우(Maximum Flow)

- 잔여 용량(Residual capacity)
  - 플로우 네트워크의 간선은 간선의 용량에서 간선의 플로우를 뺀 것에 해당하는 양만큼 플로우를 추가로 허용 가능
  - 플로우 네트워크 $G=(V, E)$에 대해서 잔여 용량(residual capacity) $C_f(u, v)$ 는 아래와 같이 정의할 수 있음

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & (u,v) \in E \\ f(v,u) & (v,u) \in E \\ 0 & \text{그 외} \end{cases}$$

  - 예)



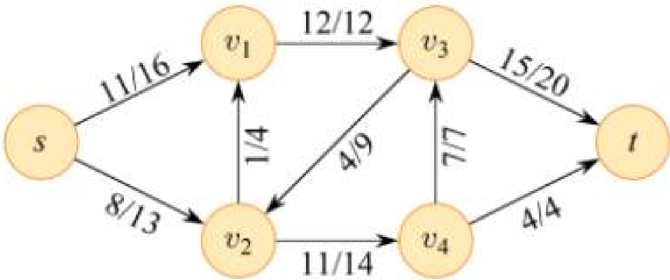
$c(s, v_1) = 16$이고 $f(s, v_1) = 11$ 이면

1) $c_f(s, v_1) = 5 \rightarrow f(s, v_1)$는 5단위만큼 더 증가 가능
2) $c_f(v_1, s) = 11 \rightarrow v_1$에서 s로 11단위만큼 되돌아 갈 수 있음

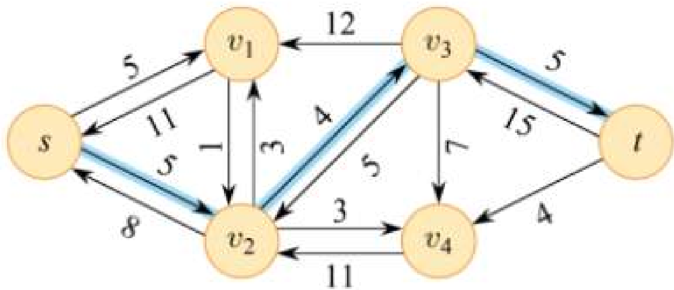
최대 플로우(Maximum Flow)

- 잔여 네트워크(Residual network)
  - 플로우 네트워크 G=(V, E)와 플로우 f에 대해 f에 의해 유도된 G의 잔여 네트워크는 G_f = (V, E_f) 이고 E_f는 다음과 같음

$$E_f = \{(u, v)\ VxV : C_f(u, v) > 0\}$$



플로우 네트워크 G=(V, E)



잔여 네트워크 G_f = (V, E_f)

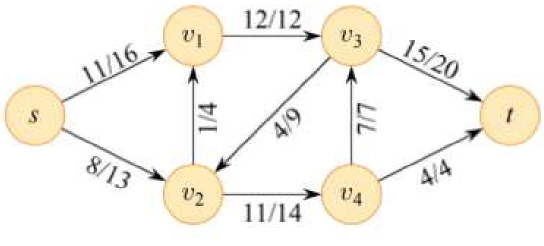
- E_f에 있는 간선들은 E에 있는 간선들이거나 그것들의 역방향 간선들이므로 다음이 성립

$$|E_f| \leq 2|E|$$

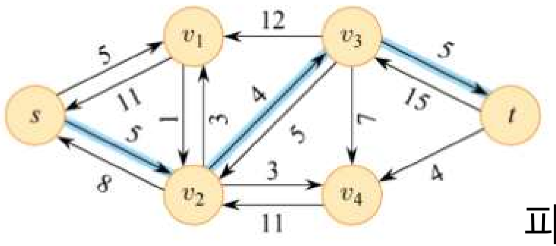
최대 플로우(Maximum Flow)

- 증강 경로(Augmenting path)
  - 잔여 네트워크 G_f의 s에서 t로의 단순 경로를 증강 경로라고 함
  - 증강 경로 p의 각 간선의 플로우를 증가시킬 수 있는 최대량인 p의 잔여 용량(residual capacity) $C_f(p)$ 은 아래와 같음

$$C_f(p) = \min\{C_f(u, v) : (u, v) \text{ is in } p\}$$

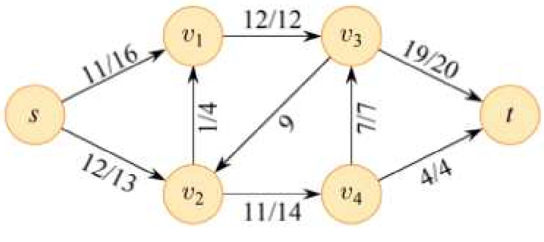


(a)

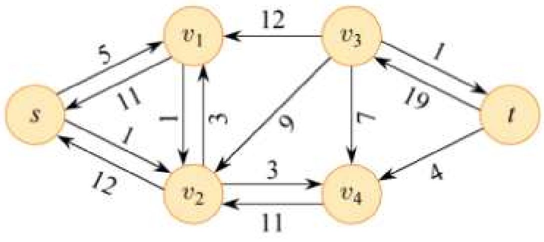


(b)

파란색 경로 : 잔여 네트워크 G_f에서의 증강 경로
잔여 용량 $C_f(p) = C_f(v_2, v_3) = 4$



(c)

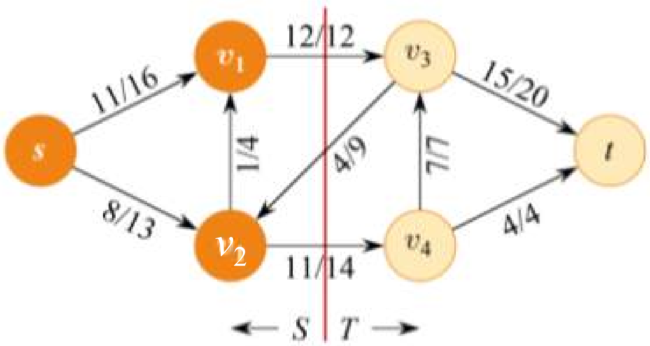


(d)

(c) 경로 p를 따라 잔여 용량 4만큼 증가시켜 얻어진 결과의 플로우
(d) (c)의 플로우에 의해 유도된 잔여 네트워크

최대 플로우(Maximum Flow)

- 플로우 네트워크의 절단(Cut)
  - 플로우 네트워크의 $G=(V, E)$의 절단 (S, T)는 V를 $s \in S$이고 $t \in T$가 되도록 하는 S와 $T=V-S$로의 분할



$S = \{s, v_1, v_2\}$, $T=\{v_3, v_4, t\}$ 인 절단 (S, T)

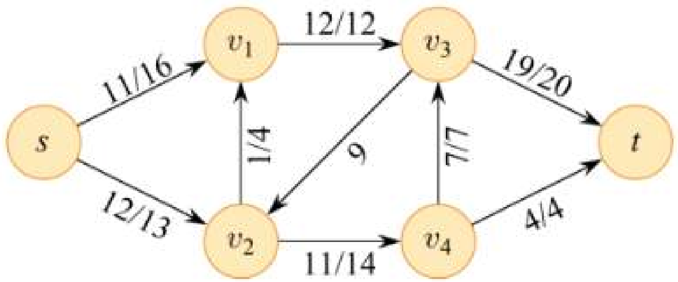
(S, T)의 순 플로우(net flow)
$f(v_1, v_3) + f(v_2, v_4) - f(v_3, v_2) = 12 + 11 - 4 = 19$

(S, T)의 용량
$c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$

  - 네트워크의 최소 절단(minimum cut)은 네트워크의 모든 절단 중 용량이 최소가 되는 절단
  - f를 출발점 s와 도착점 t를 가진 플로우 네트워크 G의 플로우라 하고 (S, T)를 G의 임의의 절단이라고 하면 (S, T)를 가로지르는 순 플로우 $f(S, T) = |f|$ 임
  - 임의의 플로우 f의 값은 G의 임의의 절단 용량에 의해 상한으로 제한됨

최대 플로우(Maximum Flow)

- Quiz
  - 아래와 같은 플로우 네트워크 G=(V, E)가 있을 때



1. **(잔여 용량(residual capacity) 구하기)** 잔여 용량 $c_f(s, v_2)$, $c_f(v_2, s)$, $c_f(v_3, t)$, $c_f(t, v_3)$를 각각 구하시오.

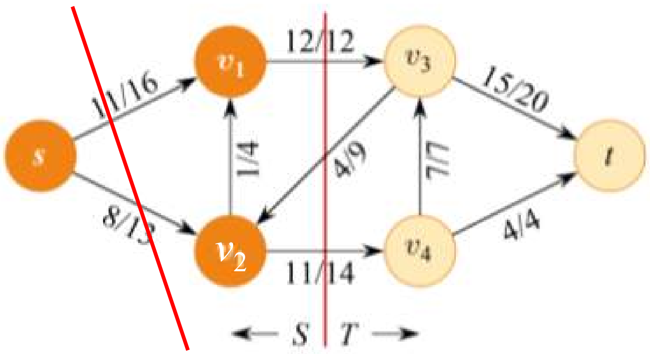
   $c_f(s, v_2) = 1$, $c_f(v_2, s) = 12$, $c_f(v_3, t) = 1$, $c_f(t, v_3) = 19$

2. **(순 플로우(net flow) 구하기)** 플로우 네트워크 G의 임의의 절단(cut)에서의 순 플로우(net flow)를 구하시오.

최대 플로우(Maximum Flow)

- 플로우 네트워크의 절단(Cut)
  - 플로우 네트워크 $G=(V, E)$의 절단 (S, T)는 V를 $s \in S$이고 $t \in T$가 되도록 하는 S와 $T=V-S$로의 분할

절단 용량 : 16 + 13 = 29

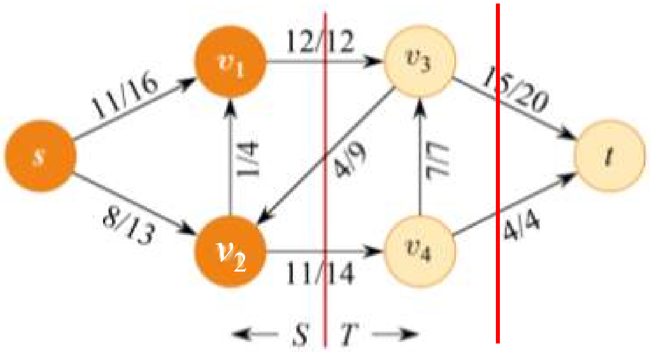


  - 네트워크의 최소 절단(minimum cut)은 네트워크의 모든 절단 중 용량이 최소가 되는 절단
  - f를 출발점 s와 도착점 t를 가진 플로우 네트워크 G의 플로우라 하고 (S, T)를 G의 임의의 절단이라고 하면 (S, T)를 가로지르는 순 플로우 $f(S, T) = |f|$ 임
  - 임의의 플로우 f의 값은 G의 임의의 절단 용량에 의해 상한으로 제한됨

최대 플로우(Maximum Flow)

- 플로우 네트워크의 절단(Cut)
  - 플로우 네트워크 $G=(V, E)$의 절단 (S, T)는 V를 $s \in S$이고 $t \in T$가 되도록 하는 S와 $T=V-S$로의 분할

절단 용량 : 20 + 4 = 24

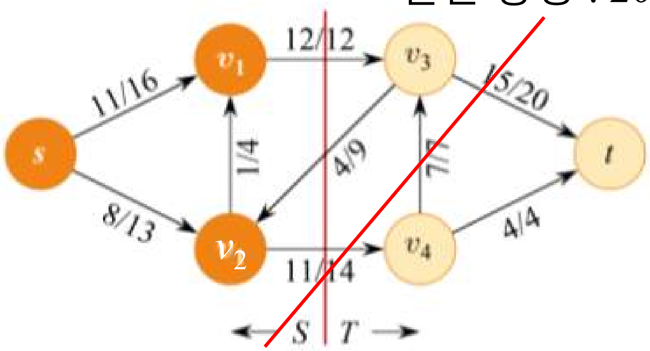


  - 네트워크의 최소 절단(minimum cut)은 네트워크의 모든 절단 중 용량이 최소가 되는 절단
  - f를 출발점 s와 도착점 t를 가진 플로우 네트워크 G의 플로우라 하고 (S, T)를 G의 임의의 절단이라고 하면 (S, T)를 가로지르는 순 플로우 $f(S, T) = |f|$ 임
  - 임의의 플로우 f의 값은 G의 임의의 절단 용량에 의해 상한으로 제한됨

최대 플로우(Maximum Flow)

- 플로우 네트워크의 절단(Cut)
  - 플로우 네트워크 $G=(V, E)$의 절단 (S, T)는 V를 $s \in S$이고 $t \in T$가 되도록 하는 S와 $T=V-S$로의 분할

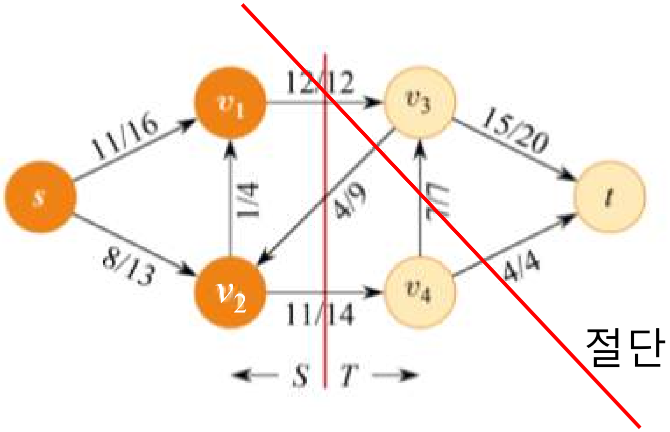
절단 용량 : 20 + 14 = 34



  - 네트워크의 최소 절단(minimum cut)은 네트워크의 모든 절단 중 용량이 최소가 되는 절단
  - f를 출발점 s와 도착점 t를 가진 플로우 네트워크 G의 플로우라 하고 (S, T)를 G의 임의의 절단이라고 하면 (S, T)를 가로지르는 순 플로우 $f(S, T) = |f|$ 임
  - 임의의 플로우 f의 값은 G의 임의의 절단 용량에 의해 상한으로 제한됨

최대 플로우(Maximum Flow)

- 플로우 네트워크의 절단(Cut)
  - 플로우 네트워크 $G=(V, E)$의 절단 (S, T)는 V를 $s \in S$이고 $t \in T$가 되도록 하는 S와 $T=V-S$로의 분할

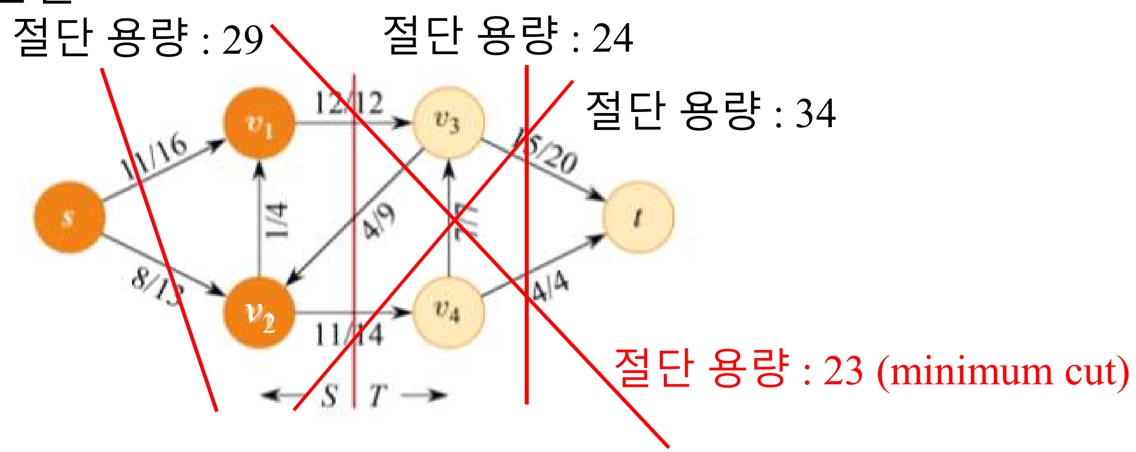


절단 용량 : 12 + 7 + 4 = 23

- 네트워크의 최소 절단(minimum cut)은 네트워크의 모든 절단 중 용량이 최소가 되는 절단
- f를 출발점 s와 도착점 t를 가진 플로우 네트워크 G의 플로우라 하고 (S, T)를 G의 임의의 절단이라고 하면 (S, T)를 가로지르는 순 플로우 $f(S, T) = |f|$ 임
- 임의의 플로우 f의 값은 G의 임의의 절단 용량에 의해 상한으로 제한됨

최대 플로우(Maximum Flow)

- 플로우 네트워크의 절단(Cut)
  - 플로우 네트워크 $G=(V, E)$의 절단 (S, T)는 V를 $s \in S$이고 $t \in T$가 되도록 하는 S와 $T=V-S$로의 분할



  - 네트워크의 최소 절단(minimum cut)은 네트워크의 모든 절단 중 용량이 최소가 되는 절단
  - f를 출발점 s와 도착점 t를 가진 플로우 네트워크 G의 플로우라 하고 (S, T)를 G의 임의의 절단이라고 하면 (S, T)를 가로지르는 순 플로우 $f(S, T) = |f|$ 임
  - 임의의 플로우 f의 값은 G의 임의의 절단 용량에 의해 상한으로 제한됨

최대 플로우(Maximum Flow)

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리
  - f 가 출발점 s와 도착점 t를 가지는 플로우 네트워크 $G=(V, E)$의 하나의 플로우라고 할 때 다음 조건들은 동등하다.
    1. f 는 G에서 최대 플로우다.
    2. 잔여 네트워크 G_f는 증강 경로를 포함하지 않는다.
    3. G의 어떤 절단 (S, T)에 대해 $|f| = c(S, T)$ 이다.

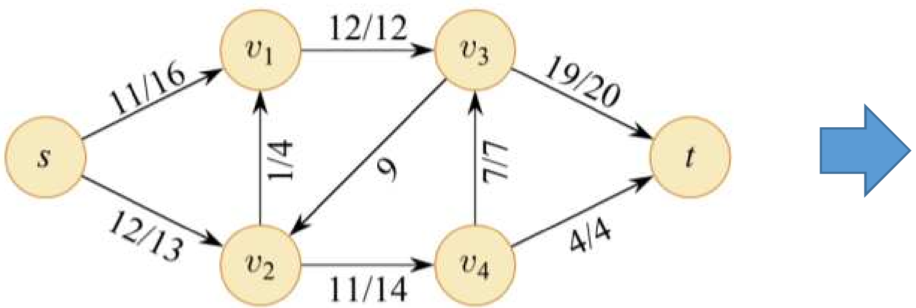
  - 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 의미
    - 최대 플로우값은 최소 절단의 용량과 같음

최대 플로우(Maximum Flow)

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리
  - f가 출발점 s와 도착점 t를 가지는 플로우 네트워크 $G=(V, E)$의 하나의 플로우라고 할 때 다음 조건들은 동등하다.
    1. f는 G에서 최대 플로우다.
    2. 잔여 네트워크 G_f는 증강 경로를 포함하지 않는다.
    3. G의 어떤 절단 (S, T)에 대해 $|f| = c(S, T)$ 이다.

  - 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 의미
    - 최대 플로우값은 최소 절단의 용량과 같음

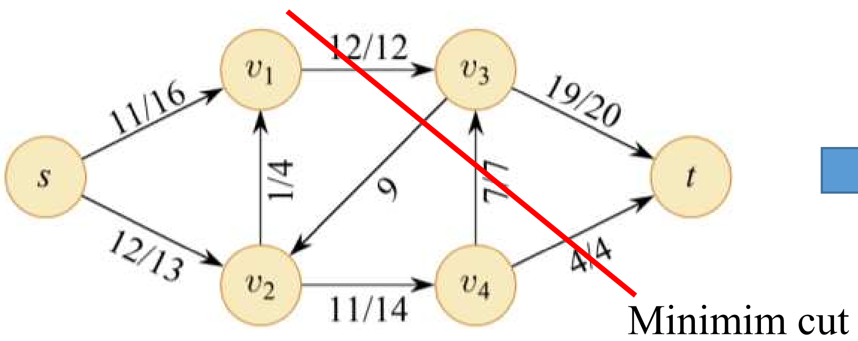
  - 예)



최대 플로우 : 23

최대 플로우(Maximum Flow)

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리
  - f가 출발점 s와 도착점 t를 가지는 플로우 네트워크 $G=(V, E)$의 하나의 플로우라고 할 때 다음 조건들은 동등하다.
    1. f는 G에서 최대 플로우다.
    2. 잔여 네트워크 G_f는 증강 경로를 포함하지 않는다.
    3. G의 어떤 절단 (S, T)에 대해 $|f| = c(S, T)$ 이다.
  - 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 의미
    - 최대 플로우값은 최소 절단의 용량과 같음
  - 예)



최대 플로우 : 23
최소 절단 용량: 12 + 0 + 7 + 4 = 23

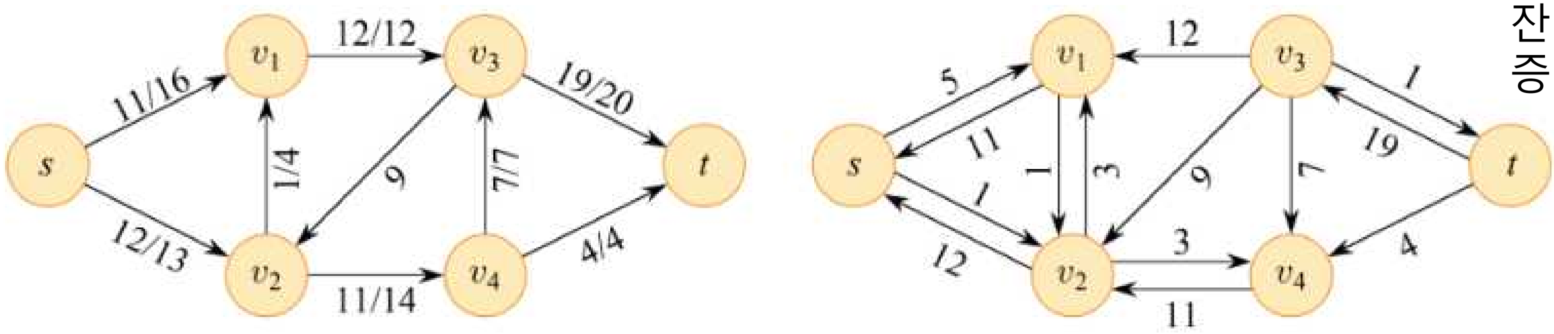
최대 플로우 = 최소 절단 용량

최대 플로우(Maximum Flow)

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리
  - f 가 출발점 s와 도착점 t를 가지는 플로우 네트워크 G=(V, E)의 하나의 플로우라고 할 때 다음 조건들은 동등하다.
    1. f 는 G에서 최대 플로우다.
    2. 잔여 네트워크 G_f는 증강 경로를 포함하지 않는다.
    3. G의 어떤 절단 (S, T)에 대해 $|f| = c(S, T)$ 이다.

  - 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 의미
    - 최대 플로우값은 최소 절단의 용량과 같음

  - 예)



잔여 네트워크의
증강 경로가 없음

최대 플로우(Maximum Flow)

- 포드-풀커슨(Ford–Fulkerson) 방법
  - 최대 플로우 문제를 풀기 위한 방법 중 하나

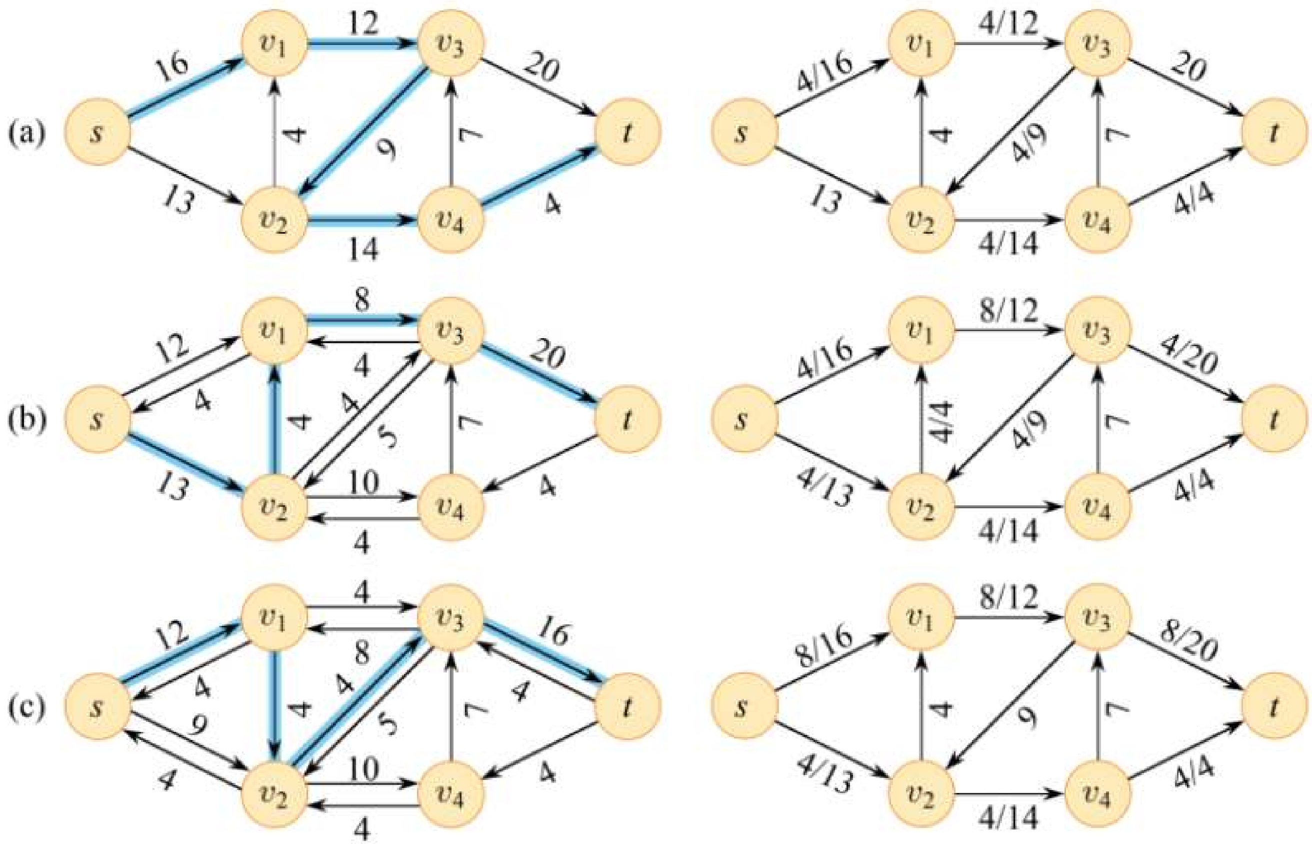
FORD-FULKERSON-METHOD(G, s, t)

플로우 f를 0으로 초기화한다.
while 잔여 네트워크 G_f에 증강 경로 p가 존재한다.
    p를 따라서 플로우 f를 증가시킨다.
return f

- 잔여 네트워크가 증강 경로를 더 이상 갖지 않을 때까지 해당 플로우를 반복해서 증가
- 작업이 끝나면 최대 플로우 최소 절단 정리가 최대 플로우를 만드는 것을 보여줌

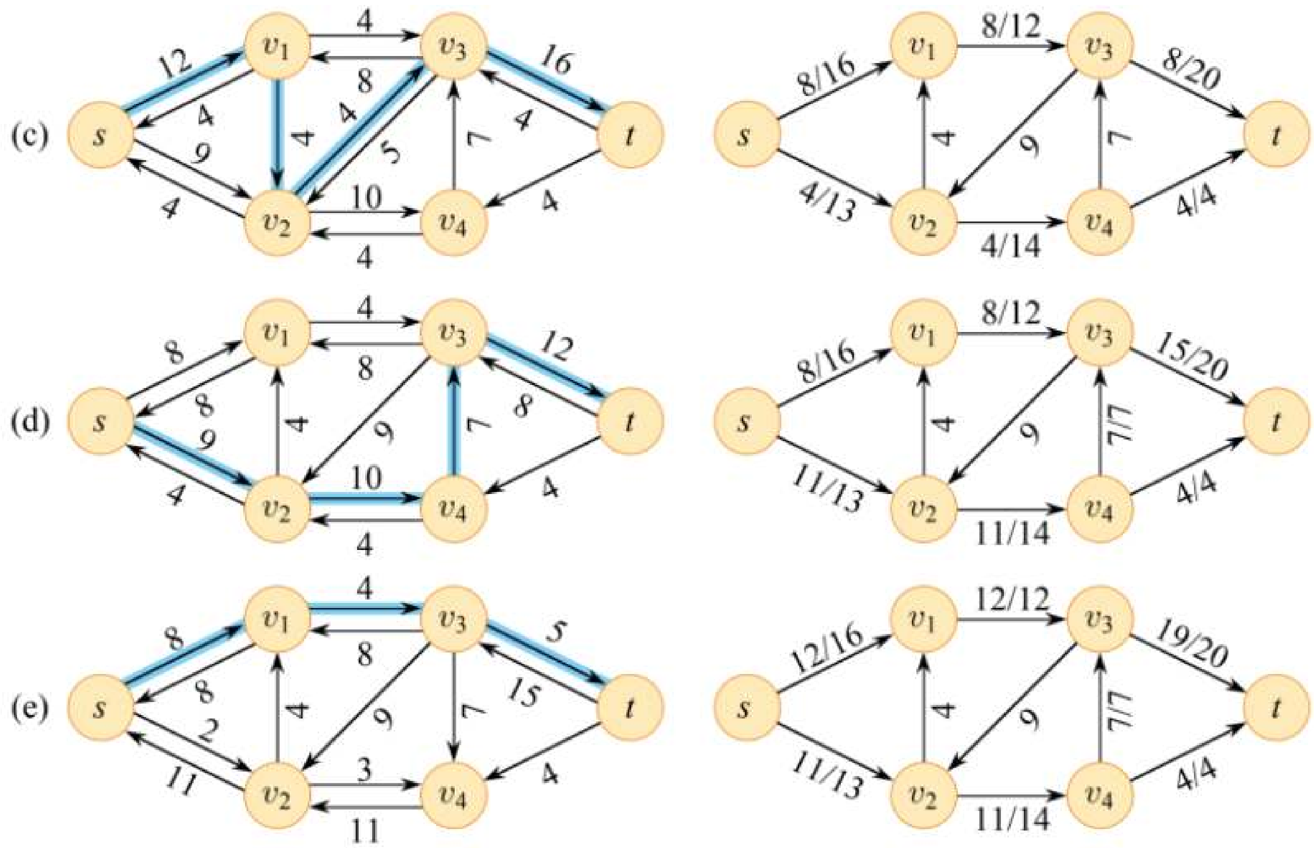
최대 플로우(Maximum Flow)

- 기본 포드-풀커슨 알고리즘



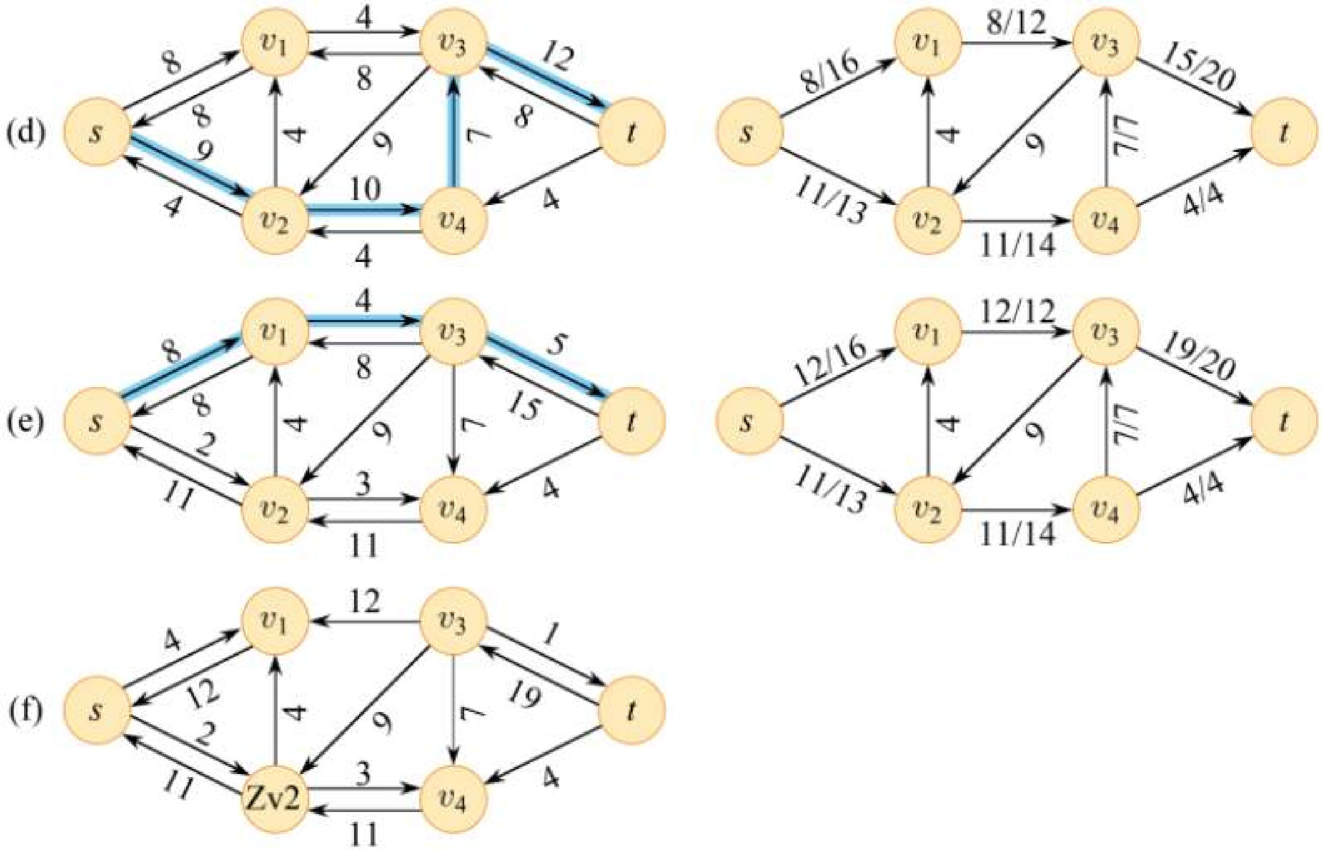
최대 플로우(Maximum Flow)

- 기본 포드-풀커슨 알고리즘



최대 플로우(Maximum Flow)

- 기본 포드-풀커슨 알고리즘



최대 플로우(Maximum Flow)

- 기본 포드-풀커슨 알고리즘
  - 기본 포드-풀커슨 알고리즘 분석

```
FORD-FULKERSON(G, s, t)
for 각 간선 (u, v) ∈ G.E
    (u, v).f = 0
while 잔여 네트워크 G_f에 s에서 t로 가는 경로 p가 존재한다.
    c_f(p) = min{c_f(u, v) : (u, v)p에 있다.}
    for p에 있는 각 간선 (u, v)
        if (u, v) ∈ G.E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
return f
```

최적의 플로우값을 $|f^*|$라고 하면

- while 루프를 최대 $|f^*|$번 수행
- 각 while 루프 반복에 $O(E)$시간이 걸림

총 수행시간 : $O(E|f^*|)$

최대 플로우(Maximum Flow)

- 기본 포드-풀커슨 알고리즘
  - 에드몬드-카프 알고리즘
    - 잔여 네트워크에서 증강 경로를 찾기 위해서 너비 우선 탐색을 이용
    - 너비 우선 탐색을 이용하므로 간선 수 기준 항상 가장 짧은 경로를 선택하게 됨
    - 총 수행시간은 $O(VE^2)$ 가 됨. 즉, 최대 플로우값과 상관없는 다항 수행 시간이 걸림

최대 플로우

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 실제 적용 예시
  - 산업·IT·영상· 네트워크 등 많은 분야에서 핵심적으로 쓰이는 알고리즘 중 하나임
    - 최대 이분 매칭 문제
      - 사람-일자리 매칭, 학생-프로젝트 배정, user-item 연결(추천시스템) 등
    - 네트워크 설계/통신
      - 인터넷 트래픽 라우팅, 데이터 전송량 최적화 등
    - 교통/물류 네트워크
      - 도로망, 파이프라인, 물류 경로 설계 등
    - 컴퓨터 비전/영상 처리
      - 이미지 분할 등
    - 소셜 네트워크 분석
      - 커뮤니티 탐색, 정보 확산 차단 등
    - 전력망/유체 흐름
      - 전력 송전망 설계, 유류/가스 배관 네트워크

최대 플로우(Maximum Flow)

- Quiz
  1. 어떤 플로우 네트워크를 절단하였을 때의 절단의 용량이 300이고 그때의 순플로우는 100이었다. 그러면 이 플로우 네트워크의 최소 절단의 용량은 100과 300 사이에 있다고 말할 수 있다.

     (O, X)

  2. 포드-풀커슨 방법은 증가 경로가 더 이상 없을 때 종료하고 이때 최대 플로우를 갖는다.

     (O, X)

  3. 기본 포드-풀커슨 알고리즘은 모든 용량이 정수들이고 최적의 플로우값이 작은 경우에 적당하다.

     (O, X)

최대 플로우(Maximum Flow)

- Quiz
  1. 어떤 플로우 네트워크를 절단하였을 때의 절단의 용량이 300이고 그때의 순플로우는 100이었다. 그러면 이 플로우 네트워크의 최소 절단의 용량은 100과 300 사이에 있다고 말할 수 있다.

     (O, X)

  2. 포드-풀커슨 방법은 증가 경로가 더 이상 없을 때 종료하고 이때 최대 플로우를 갖는다.

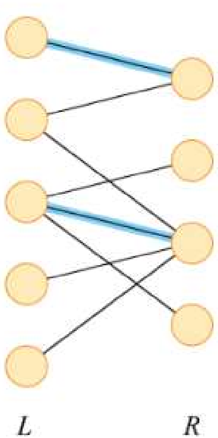
     (O, X)

  3. 기본 포드-풀커슨 알고리즘은 모든 용량이 정수들이고 최적의 플로우값이 작은 경우에 적당하다.

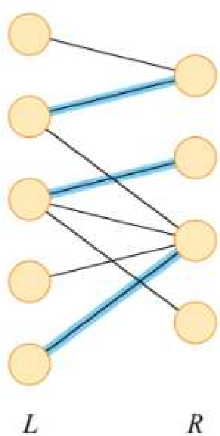
     (O, X)

최대 플로우

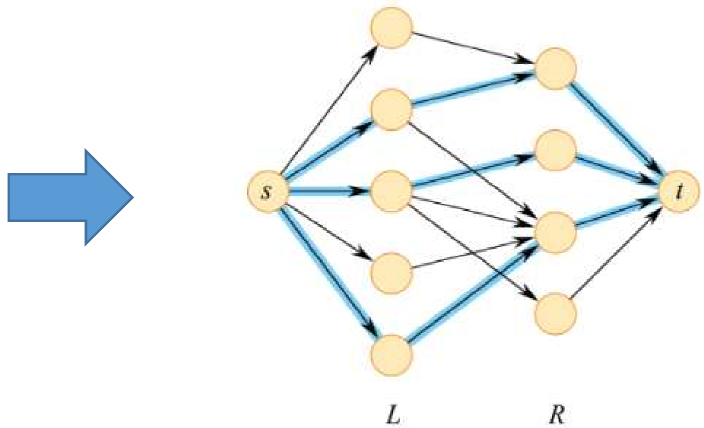
- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 실제 적용 예시
  - 최대 이분 매칭 (Maximum Biparity Matching)
    - 이분 그래프는 정점 집합이 $V = L \cup R$로 나누어지고 L과 R은 서로 겹치는 원소가 없으며 E에 있는 모든 간선은 L과 R을 연결
    - V의 모든 정점은 최소한 한 개의 부속되는 간선을 가짐
    - 다중-출발점, 다중-도착점을 갖는 최대 플로우 문제로 변환 가능



크기 2를 매칭



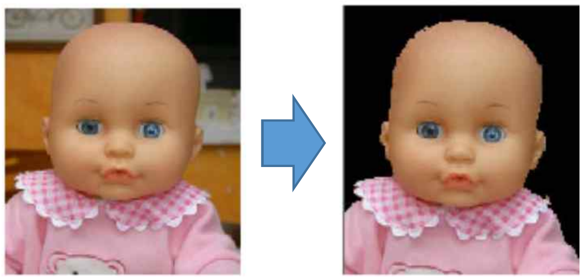
크기 3을 매칭
(최대 매칭)



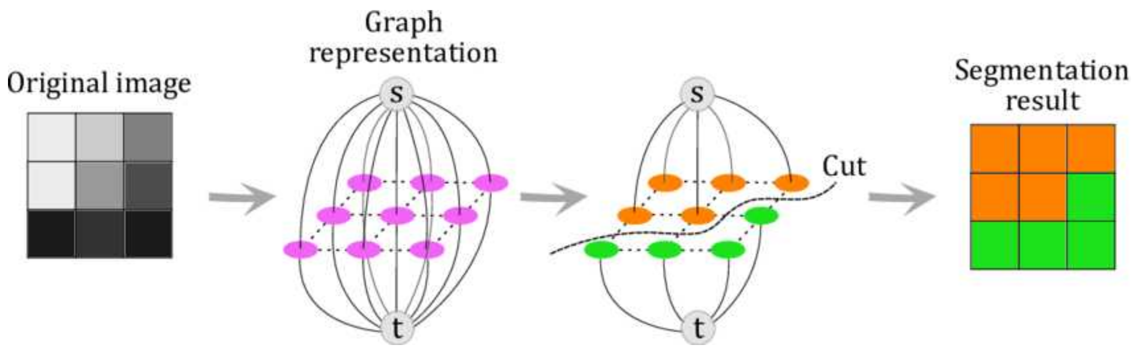
최대 플로우를 표시한
해당하는 플로우 네트워크

최대 플로우

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 실제 적용 예시
  - 컴퓨터 비전/영상 처리
    - 영상 분할
      - 영상은 많은 픽셀(Pixel)로 구성되어 있고 픽셀 간에 유사도나 이웃 관계가 존재
      - 어떤 픽셀이 객체(Foreground)에 속하고 어떤 픽셀이 배경(Background)에 속할 것인가를 결정하는 게 영상 분할 문제
      - 각 픽셀을 정점으로 하고 인접한 픽셀사이에 간선을 둠
      - 픽셀 사이의 컬러 유사도 등으로 간선에 가중치를 부여
      - 최대 플로우 최소 절단 알고리즘으로 영상을 분할

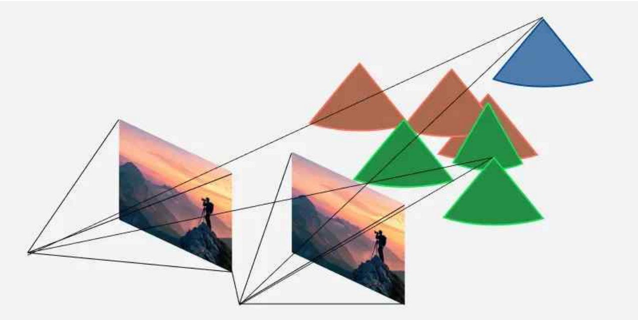


영상 분할을 통한 객체 분리의 예



최대 플로우

- 최대 플로우 최소 절단 (Maximum Flow Minimum Cut) 정리의 실제 적용 예시
  - 컴퓨터 비전/영상 처리
    - 깊이정보 추정
      - 영상의 각 픽셀이 카메라로부터 얼마나 떨어져 있는지를 나타낸 거리 지도를 깊이지도 (depth map)라고 함
      - 즉, 깊이지도에서는 각 픽셀마다 깊이정보(거리, z-값)를 가짐
      - 스테레오 카메라(2대의 카메라) 등으로 깊이 추정 가능
      - 최대 플로우 최소 절단 알고리즘을 통해 각 픽셀마다 최적의 깊이값을 추정함으로써 깊이지도를 보다 더 선명하고 오류가 적게 만들 수 있음



스테레오 카메라로 깊이정보 추정(스테레오비전)의 예시

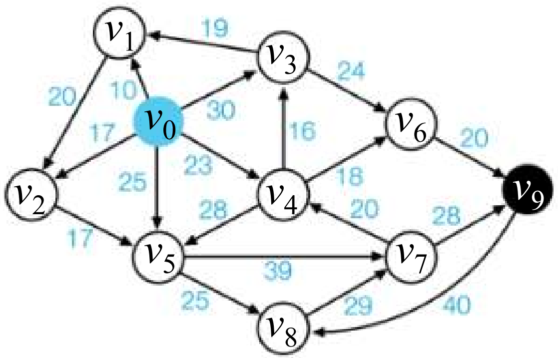
컬러영상

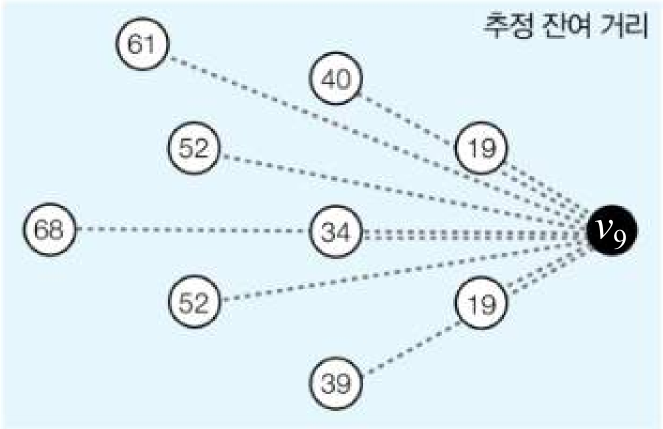
깊이지도

과제 #4

- 다익스트라 알고리즘 구현(1점 (미완성 0.2점, 오작동 0.5 점))
  - 다익스트라 알고리즘을 구현하고 아래 그래프의 출발점(v_0)부터 도착점(v_9)까지의 최단 경로 및 그때의 경로의 가중치를 구하시오.

    예) 최단 경로 : <v_0,…, v_9>, 경로의 가중치 : ??
- A* 알고리즘 구현(1점 (미완성 0.2 점, 오작동 0.5 점))
  - A* 알고리즘을 구현하고 아래 그래프의 출발점(v_0)부터 도착점(v_9)까지의 최단 경로 및 그때의 경로의 가중치를 구하시오.





과제 #4

- 제출 방법 : LMS로 노트북 파일 제출
- 코랩에서 다음과 같은 이름을 가진 노트북 파일을 생성한다.

  Algorithms_4thHW_Class본인소속반_학번_영문이름.ipynb

  예) Algorithms_4thHW_Class1_23xxxxx_HanshinLim.ipynb
- 생성된 코랩 노트북 파일(.ipynb)에서 코드를 구현 및 결과를 출력한다. 결과 출력 시 어떤 결과인지를 명시한다.
- Due : 11월 28일 밤 12시 (이후 제출 0점)

내용 정리

- 단일 출발점 최단 경로 알고리즘
- 최대 플로우 최소 절단 정리 및 최대 플로우 알고리즘