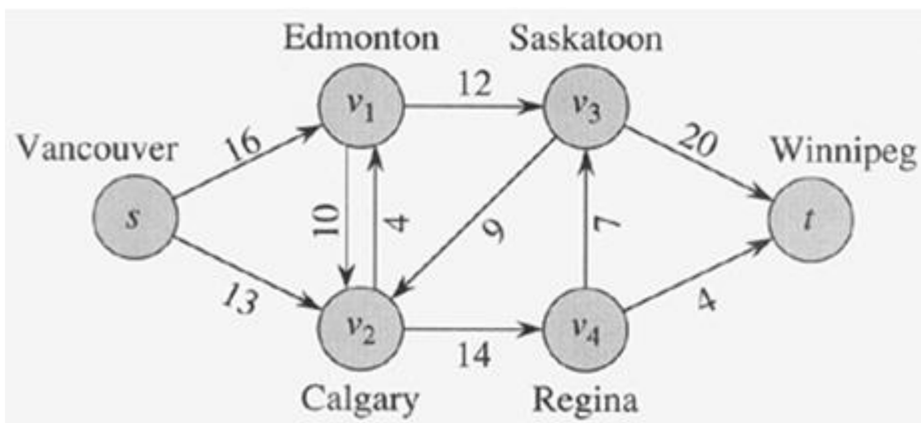
Maximum Flow

Flow Networks

- **Flow Networks** G=(V,E)
  - 각 edge (u,v) 가 capacity c(u,v)≥ 0 를 갖는 directed graph.
  - one source (s), one sink (t)
  - 모든 vertex v ∈ V 에 대하여, path s → v → t 존재 (connected)
  - |E| ≥ |V| -1



Flow Networks

- Flow f(u,v)
  - edge (u,v) 를 통과하는 양

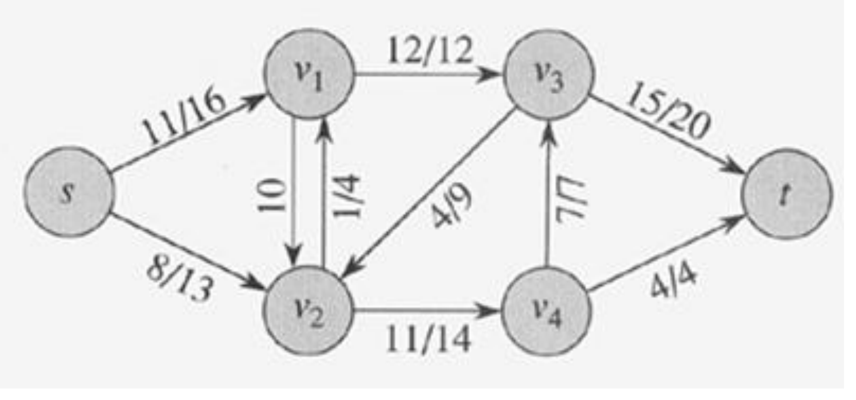
(i) $f(u,v) \le c(u,v)$, ∀u,v ∈ V (capacity constraint)

(ii) $f(u,v) = -f(v,u)$, ∀u,v ∈ V (skew symmetry)

(iii) $\sum_{v \in V} f(u,v) = 0$,∀u ∈ V- {s,t} (flow conservation)

- value of flow
  - source 에서 방출되는 flow 의 합 $|f| = \sum_{v \in V} f(s,v)$
- maximum-flow problem

maximum value of flow 를 찾는 문제



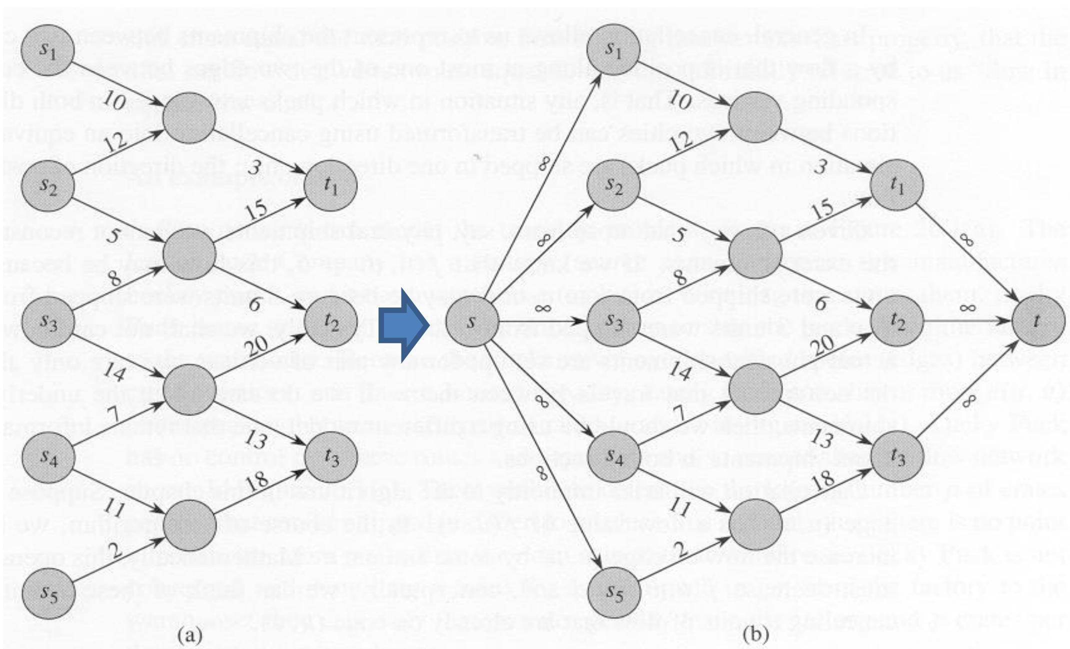
Flow Networks

- Networks with multiple sources and sinks
  - M 개의 sources $\{s_1, s_2, \ldots, s_m\}$ 와 n 개의 sinks $\{t_1, t_2, \ldots, t_n\}$ 을 갖는 max-flow problem
  - 1 개의 source 와 1개의 sink 를 갖는 표준 문제로 변환

    1) super-source s 와 super-sink t 추가

    2) $c(s, si) = \infty$, $i=1,\ldots,m$,

    $c(ti, t) = \infty$, $i=1,\ldots,n$



Ford-Fulkerson Method

- Residual capacity

$$c_f(u,v) = c(u,v) - f(u,v)$$

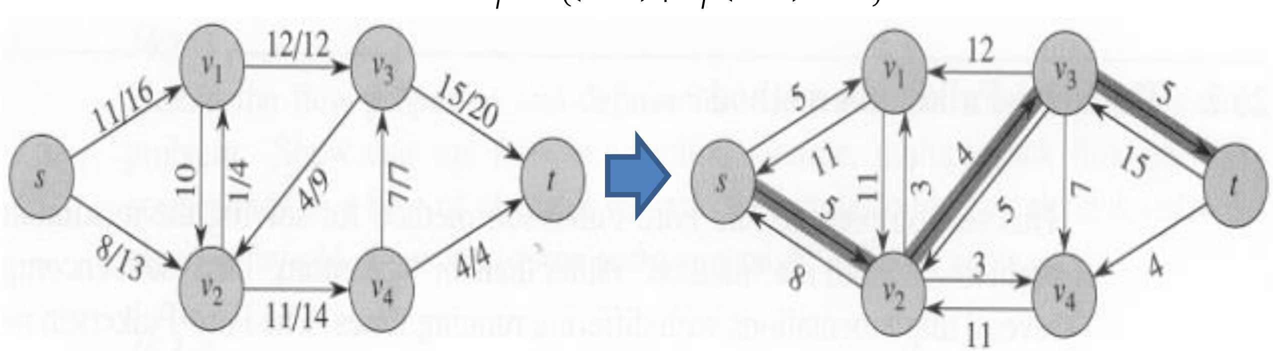
  – Edge (u,v) 에 대한 잔여 허용량 또는 추가 가능한 flow

(ex) $c(u,v)=16,\ f(u,v)=11 \ \rightarrow\ c_f(u,v)=5$

$c(u,v)=16,\ f(u,v)=-4 \ \rightarrow\ c_f(u,v)=20$

- Residual network
  – Residual edge 로 구성된 network

$$G_f = (V, E_f)$$
$$E_f = \{(u,v) \mid c_f(u,v) > 0\}$$



(residual network)

Ford-Fulkerson Method

- Augmenting Path (p)
  - residual network G_f 에서, s 부터 t 까지의 path
  - 모든 edge 는 positive flow

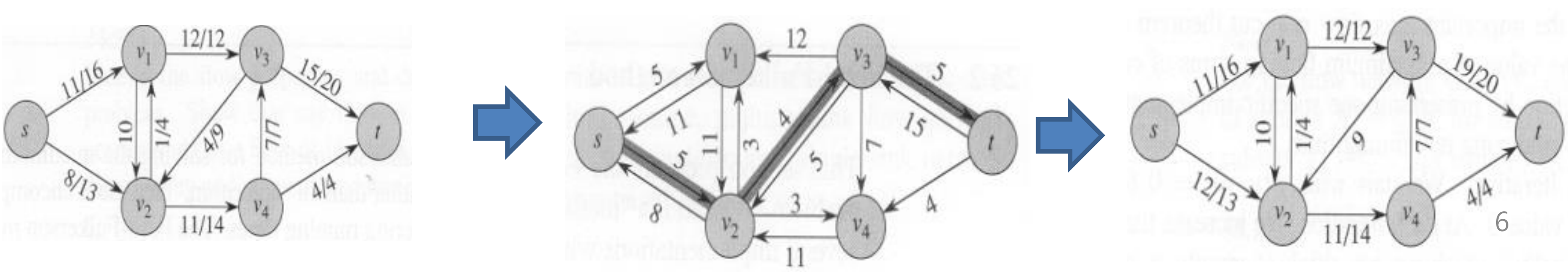
- Residual capacity of augmenting path

$$c_f(p) = \min\{c_f(u,v) \mid (u,v) \text{ is on } p\}$$

- New flow

$$f' = f + f_p$$

$$f_p = \begin{cases} c_f(p) & ,(u,v) \text{ is on } p \\ -c_f(p) & ,(v,u) \text{ is on } p \\ 0 & ,\text{otherwise} \end{cases}$$



Ford-Fulkerson Method

- Max-flow min-cut theorem

> f is a maximum flow in G
> $\leftrightarrow$ residual network G_f contains no more augmenting paths

- Basic Algorithm

```
FORD-FULKERSON-METHOD(G, s, t)
initialize flow f to 0
while there exists an augmenting path p
      do augment flow f along p
return f
```

Ford-Fulkerson Method

- Ford-Fulkerson (G,s,t)

FORD-FULKERSON(G, s, t)

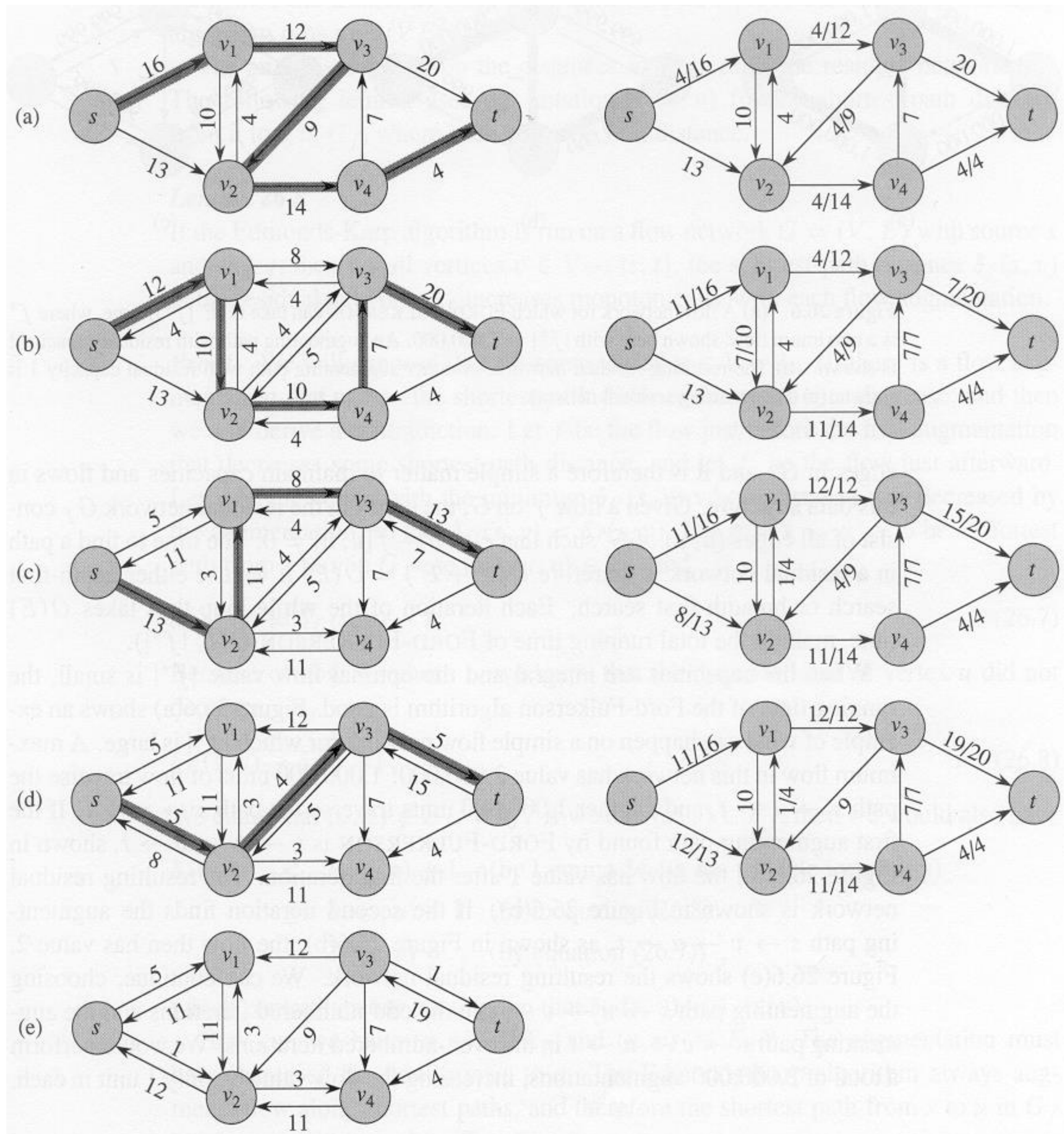
1 **for** each edge $(u, v) \in E[G]$
2   **do** $f[u, v] \leftarrow 0$
3         $f[v, u] \leftarrow 0$
4 **while** there exists a path p from s to t in the residual network G_f
5   **do** $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
6   **for** each edge (u, v) in p
7   **do** $f[u, v] \leftarrow f[u, v] + c_f(p)$
8                 $f[v, u] \leftarrow -f[u, v]$

Running Time: O(E |f*|)

Line 1-3: Q(E)
while loop (line 4-8) 반복횟수: 최대 |f*| 번 (1씩 증가될 수 있으므로)
while loop 내의 G_f 에서 s->t path 찾기: O(E)
- breadth-first search or depth-first search

Ford-Fulkerson Method

- Operations



Ford-Fulkerson Method

(Q)

Ford-Fulkerson Method

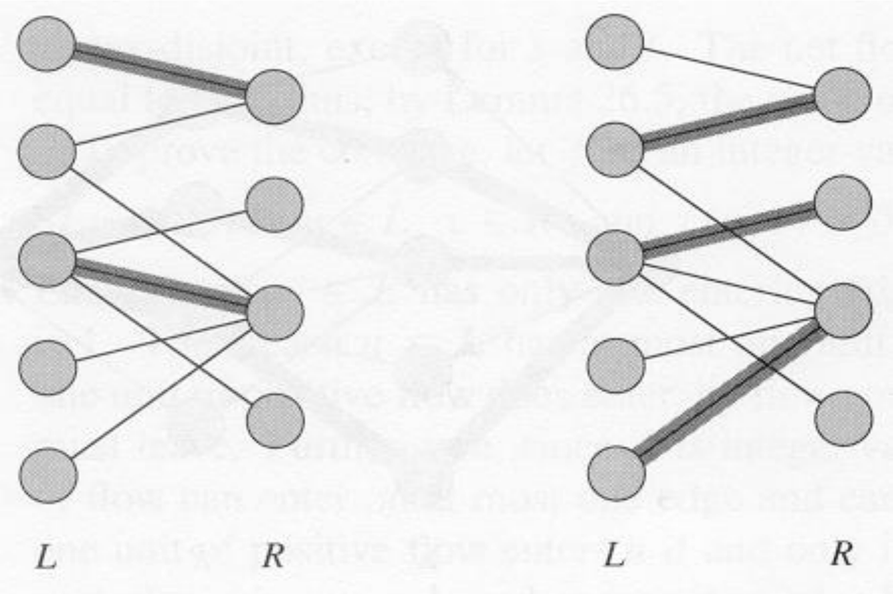
- Edmonds-Karp Algorithm
  - Ford-Fulkerson algorithm 개선
  - residual network 에서 augmenting path 를 BFS(breadth-first search) 로 찾음
  - running time= O(V E^2)
    - Edmonds-Karp 알고리즘으로 수행되는 flow augmentation 의 총 수 = O(V E) (Theorem 26.9)
    - 각 augmenting path 에 대한 수행: O(E)

Maximum Bipartite Matching

- Bipartite graph
  (i) undirected graph $G = (V, E)$
  (ii) $V = V_1 \cup V_2$, 단, $V_1 \cap V_2 = \varnothing$ (partitioned)
  (iii) $(u, v) \in E$ 에 대하여,
  $u \in V_1, v \in V_2$ 또는 $u \in V_2, v \in V_1$
- Matching
  (i) $M \subseteq E$
  (ii) 모든 $v \in V$ 에 대하여, 최대 1 개의 edge 가 연결(incident on)됨
  - cardinality : matching 의 수, $|M|$
- Maximum bipartite matching
  – Bipartite graph 에서 최대의 cardinality 를 갖는 matching

Maximum Bipartite Matching

(ex)



(a) (b)

Maximum Bipartite Matching

- Corresponding flow network
  - G=(V, E) : bipartite graph , V = L ∪R
  - G′= (V′, E′): corresponding flow network

    V′ = V ∪ { s, t }

    E′ = { (s,u) | u ∈ L}

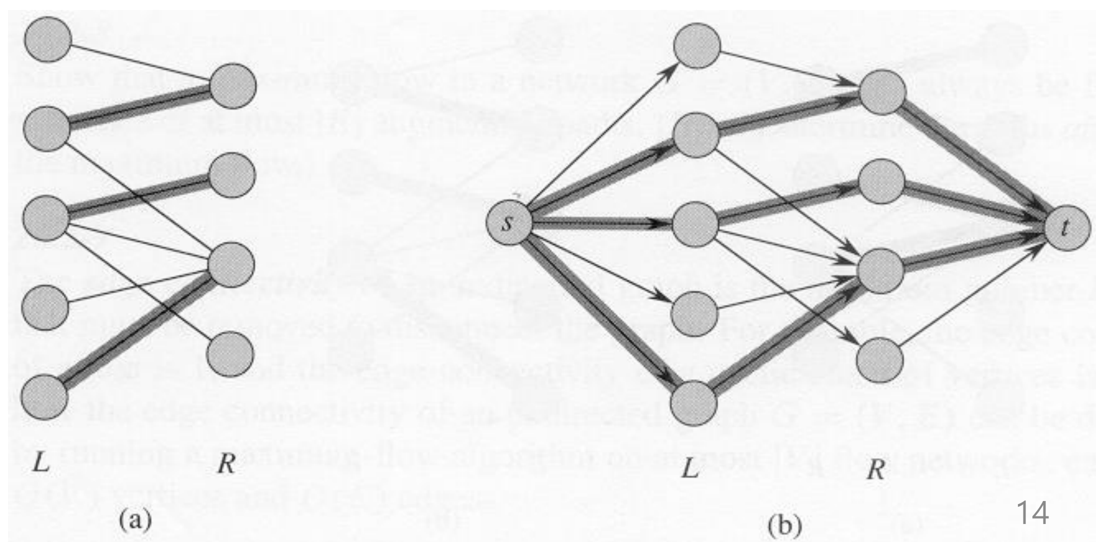
    ∪ {(u,v) | u∈ L, v ∈ R, and (u,v) ∈ E}

    ∪ {(v,t) | v ∈ R}

    c(u′,v′) = 1, ∀(u′,v′) ∈ E′
  - |M| = |f|

    M: matching in G

    f: flow in G′



Maximum Bipartite Matching

- Algorithm

S1. bipartite graph G 를 corresponding flow network G′ 로 변환
S2. G′ 에 대하여 Ford-Fulkerson 알고리즘 적용