Algorithm Analysis and Data Structures

CSCI 7432 - Fall 2022

Advanced Graph Algorithms: Flow Networks

Dr. Yao XU

Assistant Professor

Department of Computer Science

Georgia Southern University

Email: yxu@georgiasouthern.edu

Table of Contents

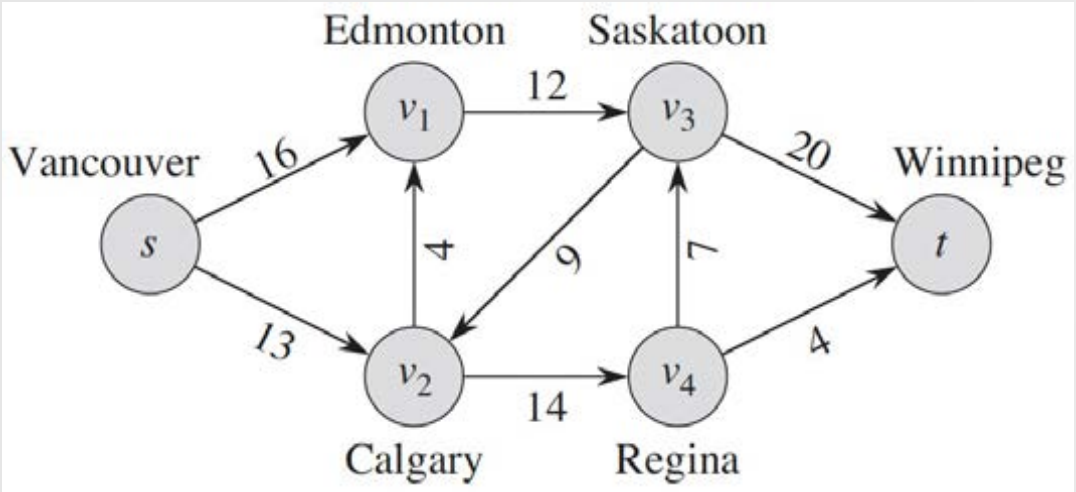
1. Flow Networks (26.1)
   - The Maximum-Flow Problem
   - The Minimum Cut Problem
2. The Ford-Fulkerson Method (26.2)
   - Residual Network
   - Augmenting Paths
   - Max-Flow Min-Cut Theorem
   - The Basic Ford-Fulkerson Algorithm
   - The Edmonds-Karp algorithm
3. Maximum Bipartite Matching (26.3)

Flow Networks

The Maximum-Flow Problem

Flow Networks

- A ***flow network*** is a directed graph $G = (V, E)$ in which
  - each edge $(u, v) \in E$ has a nonnegative ***capacity*** $c(u, v) \geq 0$;
  - if $(u, v) \notin E$, then for convenience $c(u, v) = 0$;
  - there are two distinguished vertices: a ***source*** $s \in V$ and a ***sink*** $t \in V$.
- **Intuition:** Material originates at source s and is sent to sink t.
- **Example:** A flow network for the Lucky Puck Company's trucking problem.

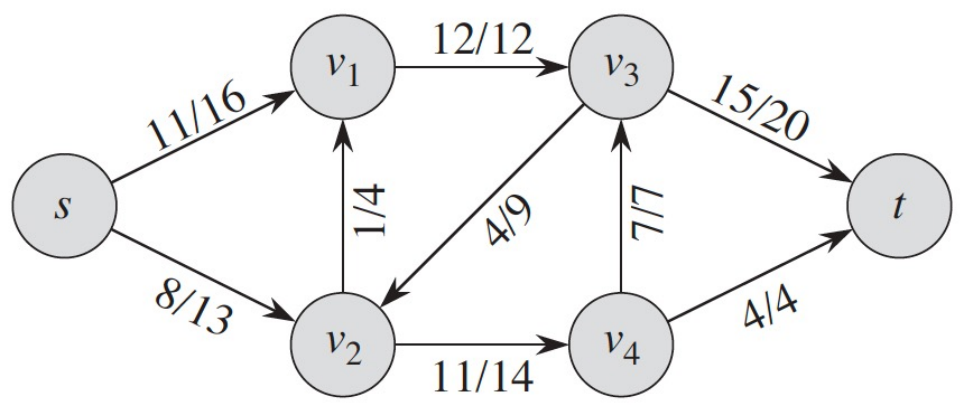


Flow

- A ***flow*** in a flow network $G = (V, E)$ is a function $f: V \times V \to \mathbb{R}$ satisfying:
  - ***Capacity constraint***: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$;
  - ***Flow conservation***: For all $u \in V - \{s, t\}$,

$$\underbrace{\sum_{v \in V} f(v, u)}_{\text{flow into } u} = \underbrace{\sum_{v \in V} f(u, v)}_{\text{flow out of } u}.$$

- ***Value*** of a flow f is: $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$.
- **Example: Q:** $|f| = ?$



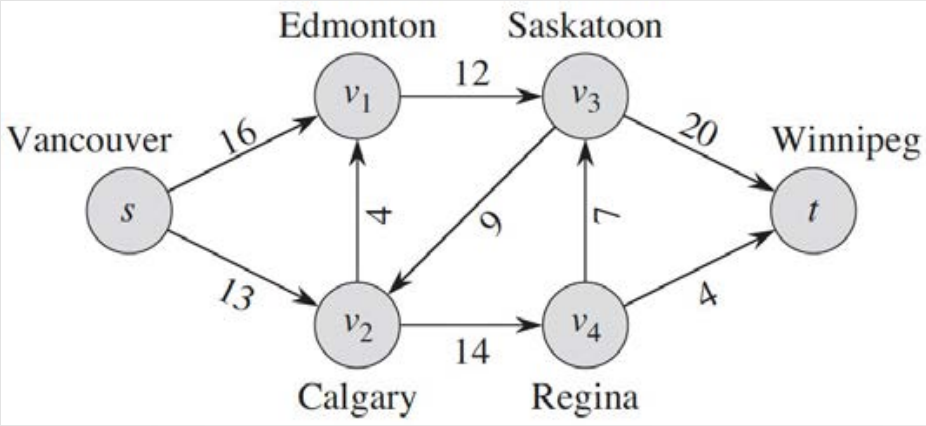
A: $|f| = 19$

The Maximum-Flow Problem

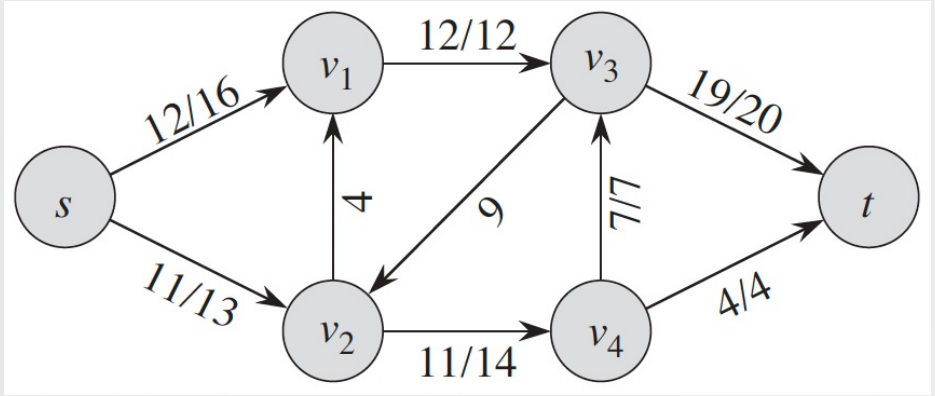
Input: A flow network $G = (V, E)$ with source $s \in V$, sink $t \in V$, and capacity $c(u, v) \geq 0$ for each edge $(u, v) \in E$.

Output: A flow of maximum value.

- **Goal:** Find flow $f(u, v)$ for each edge $(u, v) \in E$ s.t.
  - For all $(u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$ – Capacity constraint
  - For all $u \in V - \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$ – Flow conservation
  - $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$ is maximized
- **Example:**



Max flow:

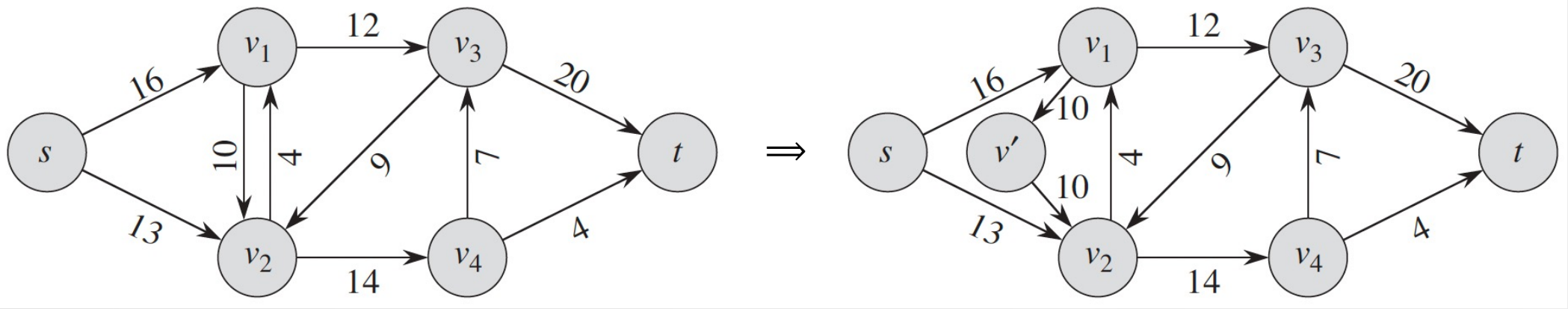


Flow Network Assumptions

Assumption 1: Self-loops are disallowed.

Assumption 2: If $(u, v) \in E$, then the reverse edge $(v, u) \notin E$.

- When both $(u, v) \in E$ and $(v, u) \in E$, which are called ***antiparallel*** edges, we transform G into an equivalent one with no antiparallel edges by choosing one of them, say (u, v),
  - Adding a new vertex v', and
  - Replacing (u, v) by (u, v') and (v', v), with $c(u, v') = c(v', v) = c(u, v)$



Flow Networks

The Minimum Cut Problem

The Minimum *s-t* Cut Problem

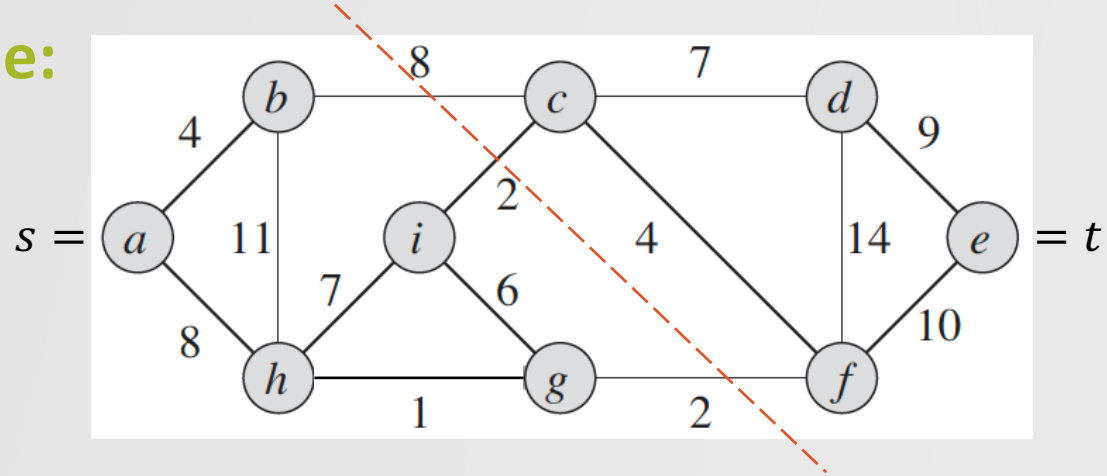
Input: An undirected graph $G = (V, E)$, two veritices $s, t \in V$, and weight $w(u, v) \geq 0$ for each edge $(u, v) \in E$.

- A s-t ***cut*** (S, T) of G is a partition of V into two sets S and $T = V - S$ such that $s \in S$ and $t \in T$.

Output: Find a s-t cut (S, T) of G with minimum weight:

$$w(S, T) = \sum_{u \in S} \sum_{v \in T} w(u, v).$$

- **Example:**



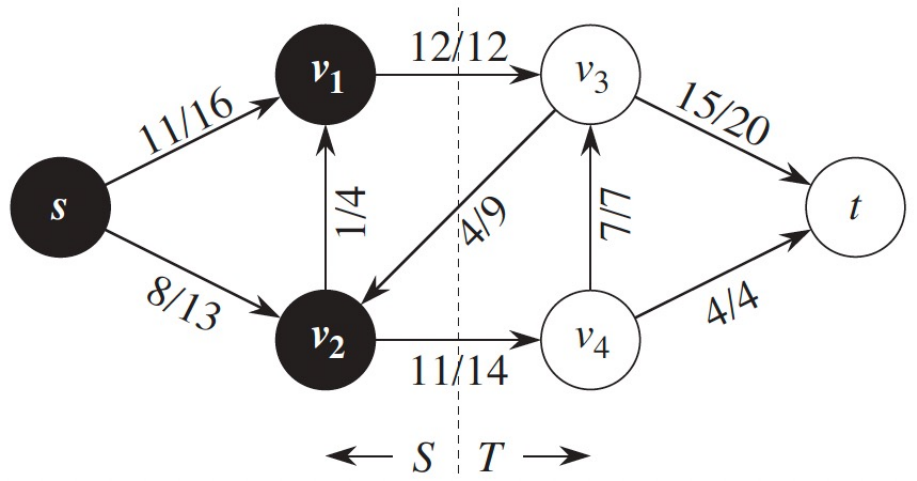
- A minimum s-t cut:
  $S = \{a, b, i, h, g\}$
  $T = \{c, d, e, f\}$
- $w(S, T) = 8 + 2 + 2 = 12$

Cuts of Flow Networks

- A ***cut*** (S, T) of a flow network $G = (V, E)$ is also a partition of V into two sets S and $T = V - \mathrm{S}$ such that $s \in S$ and $t \in T$.
- For a flow f, the ***net flow*** across cut (S, T) is

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u).$$

- **Example:** $f(S, T) = ?$



A: $f(S, T) = 19$

Net Flow Across A Cut (1/2)

For a flow f, the ***net flow*** across cut (S, T) is

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u).$$

Claim:

- $f(X,X) = 0$
- If $X \cap Y = \emptyset$, then

$$f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$$
$$f(Z, X \cup Y) = f(Z,X) + f(Z,Y)$$

Net Flow Across A Cut (2/2)

For a flow f, the ***net flow*** across cut (S, T) is

$$f(S,T) = \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u).$$

Lemma: For any flow f and any cut (S, T), we have $f(S,T) = |f|$.*

Proof. *Recall:* $|f| = \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s) = f(\{s\}, V)$

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) - 0 \\ &= f(\{s\},V) + f(S-\{s\},V) \end{aligned}$$

Due to flow conservation, $\sum_{v\in V} f(v,u) = \sum_{v\in V} f(u,v)$ for $u \in V - \{s,t\}$.

So, $f(S-\{s\},V) = 0 \Longrightarrow f(S,T) = f(\{s\},V) = |f|$ □

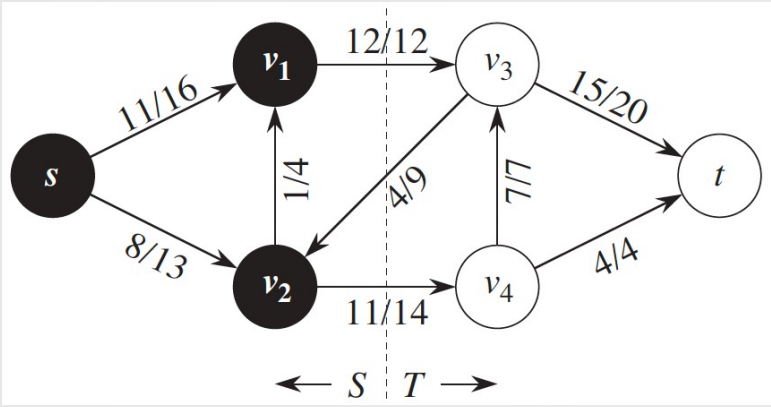
**See Lemma 26.4 on p.721-722 of the textbook for a complete proof.*

Capacity of A Cut

- The ***capacity*** of cut (S, T) is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v).$$

- **Example:** $c(S,T) = ?$



A: $c(S,T) = 26$

Corollary: The value of any flow ≤ capacity of any cut.

Proof. $|f| = f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$

$$\leq \sum_{u \in S} \sum_{v \in T} c(u,v) = c(S,T).$$

□

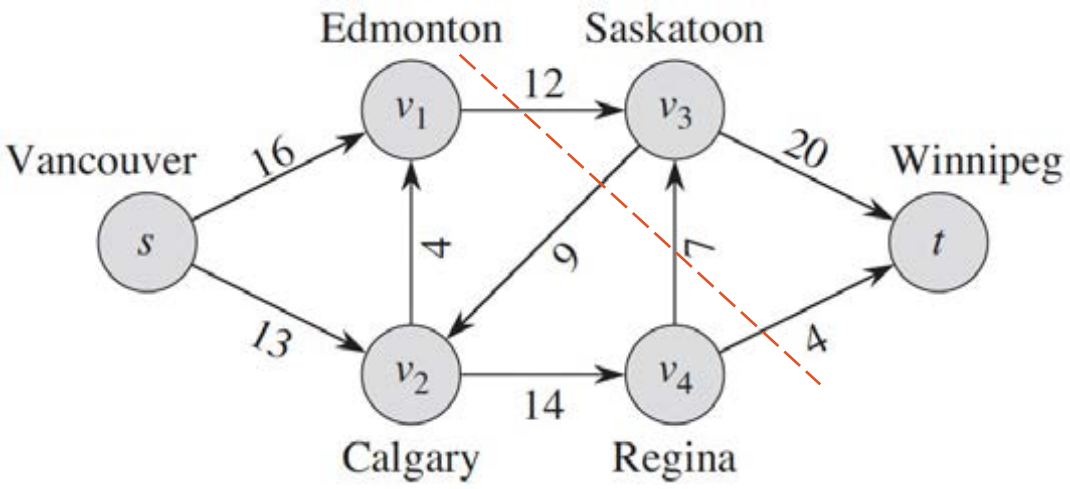
Minimum Cut of A Flow Network

Input: A flow network $G = (V, E)$ with source $s \in V$, sink $t \in V$, and capacity $c(u, v) \geq 0$ for each edge $(u, v) \in E$.

Output: A s-t cut (S, T) of G with minimum capacity:

$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v).$$

- **Example:**



- A minimum s-t cut:
  $S = \{s, v_1, v_2, v_4\}$
  $T = \{v_3, t\}$
- $c(S, T) = ?$

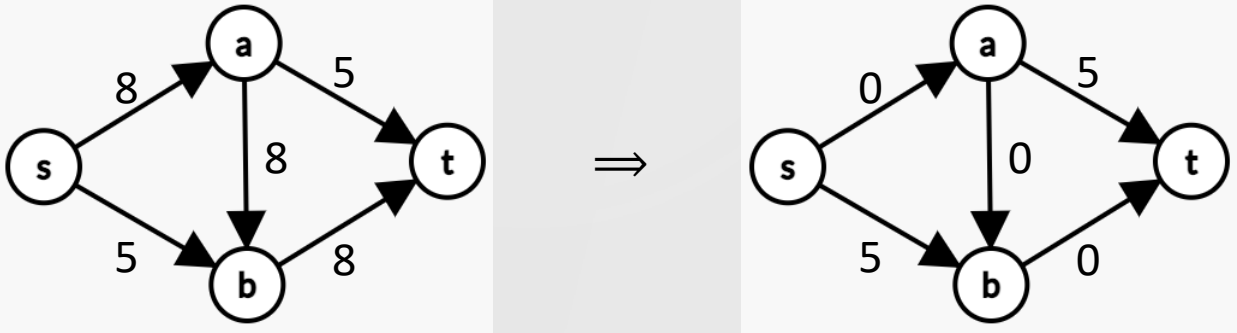
The Ford-Fulkerson Method

The Ford-Fulkerson Method

Ford-Fulkerson-Method(G, s, t)

1 initialize flow f to 0
2 **while** there exists an ***augmenting path*** p in the ***residual network*** G_f
3 augment flow f along p
4 **return** f

- The ***residual network*** consists of edges with capacities that represent how we can change the flow on edges of G.
- An ***augmenting path*** p is a simple path from s to t in the residual network.
- **Example:**



The Ford-Fulkerson Method

Residual Network

Residual Network

- Given a flow f in network $G = (V, E)$, the ***residual network*** consists of edges with capacities that represent how we can change the flow in G.
- That's the ***residual capacity***:

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

- Then, the ***residual network*** is $G_f = (V, E_f)$, where

$$E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$$

Residual Network Examples (1/2)

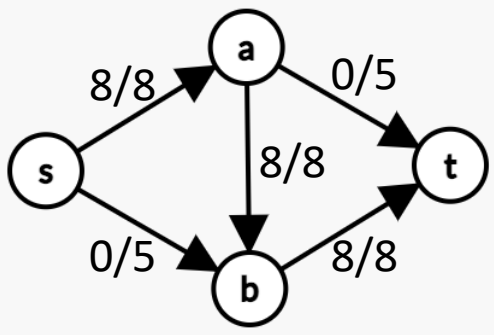
- The ***residual network*** is $G_f = (V, E_f)$, where

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}.$$

- The ***residual capacity***:

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

- **Example 1:** Given a flow f in network G as follows, what is G_f?

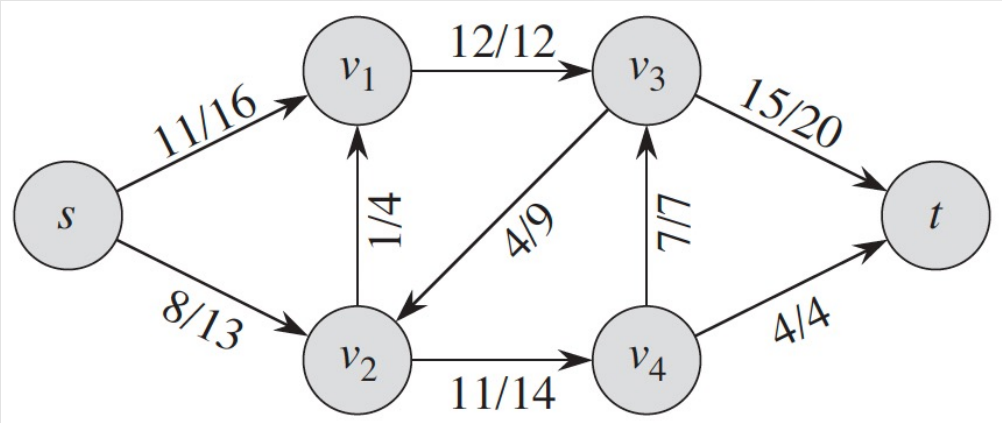


Residual Network Examples (2/2)

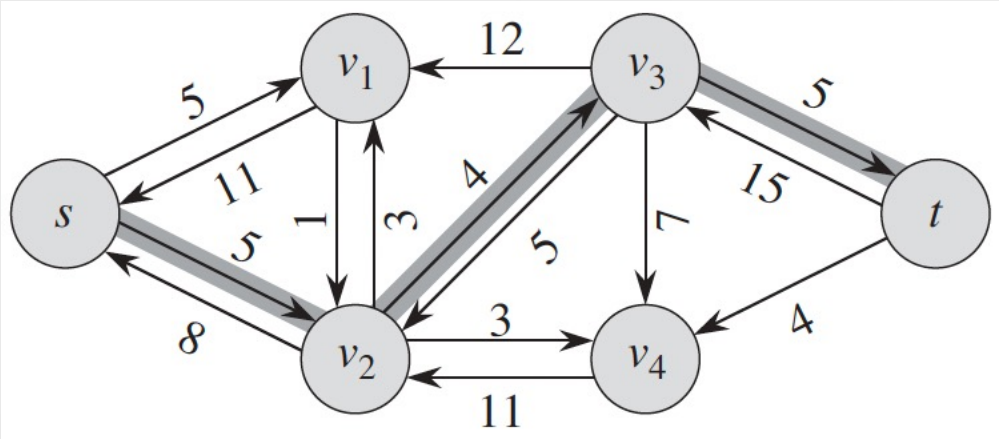
- The ***residual capacity***:

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

- **Example 2:**



G with f



G_f

The Ford-Fulkerson Method

Augmenting Path

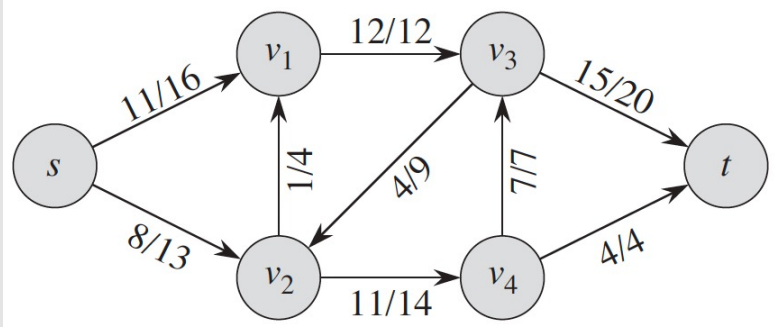
Augmenting Path

- An ***augmenting path*** p is a simple path from s to t in G_f.
- The flow value can be increased along an augmenting path p by

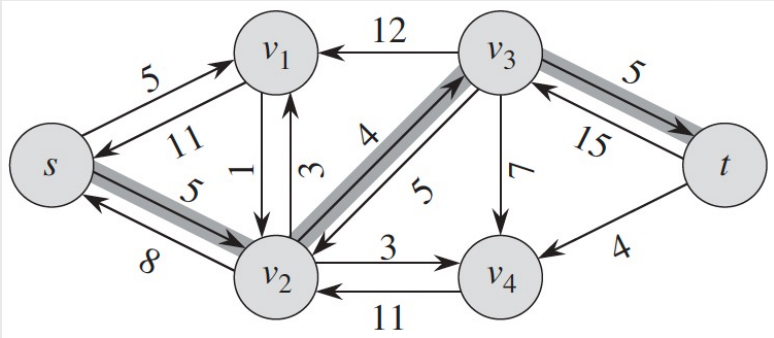
$$c_f(p) = \min_{(u,v)\in p}\{c_f(u,v)\}.$$

- **Example:**

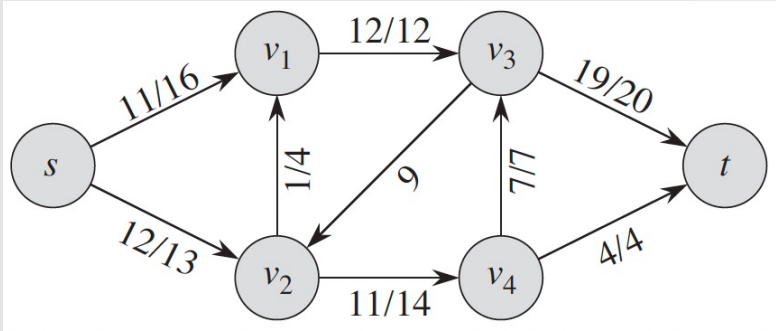
G with f_1:



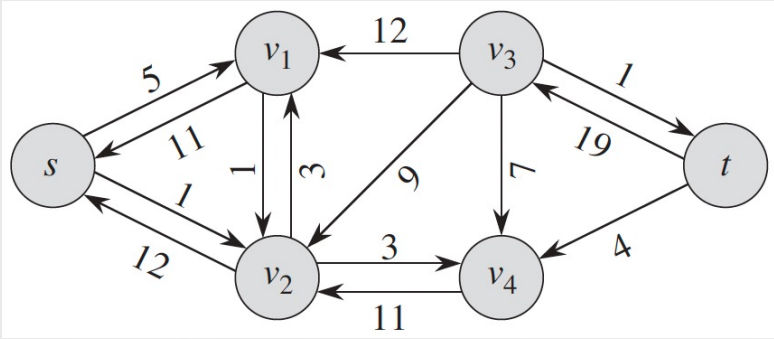
G_{f_1}:



G with f_2:



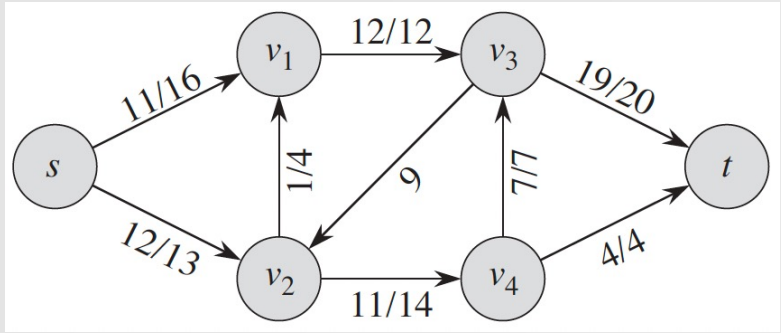
G_{f_2}:



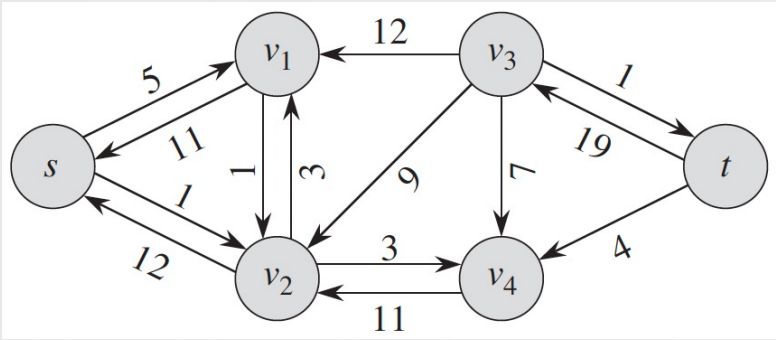
Augmented Flow Network

- When there is no augmenting path in G_f, f is a maximum flow in G.
- **Example:**

G with f_2:



G_{f_2}:



- There is no augmenting path in G_{f_2}.
  - Consider cut (S, T) in G_{f_2}: $S = \{s, v_1, v_2, v_4\}$ and $T = \{v_3, t\}$
  - No edges cross the cut in the direction from S to T.
- Thus, the maximum flow in G is f_2 and $|f_2| = 23$.

The Ford-Fulkerson Method

Max-Flow Min-Cut Theorem

Max-Flow Min-Cut Theorem (1/3)

Max-Flow Min-Cut Theorem* If f is a flow in a flow network $G = (V, E)$ with source s and sink t, the the following conditions are equivalent:

1) f is a maximum flow in G.
2) The residual network G_f contains no augmenting paths.
3) $|f| = c(S, T)$ for some cut (S, T) of G.

Proof. (1) ⇒ (2): (prove by contraposition)

- Assume there is an augmenting path in G_f.
- Then the flow value could be increased.

(3) ⇒ (1):

- $|f| \leq c(S, T)$ for any cut (S, T). - According to Corollary (on slide 12)

** This is Theorem 26.6 on p.723 of the textbook.*

Max-Flow Min-Cut Theorem (2/3)

Max-Flow Min-Cut Theorem If f is a flow in a flow network $G = (V, E)$ with source s and sink t, the the following conditions are equivalent:

2) The residual network G_f contains no augmenting paths.

3) $|f| = c(S, T)$ for some cut (S, T) of G.

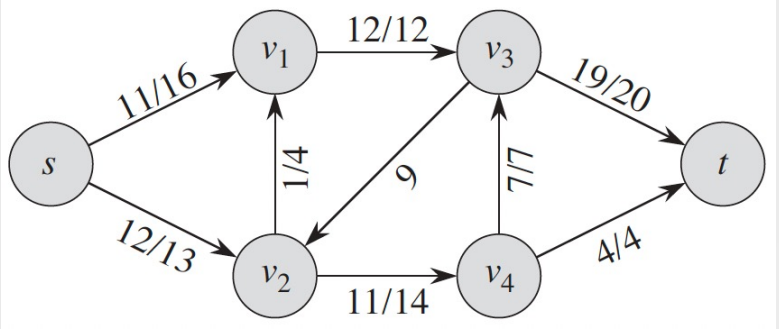
Proof. (cont'd) (2) ⇒ (3): When G_f contains no augmenting paths, define a cut (S, T) s.t.

- $S = \{v \in V : \exists\ s \rightsquigarrow v \text{ path in } G_f\}$ $(s \in S)$
- $T = V - S$ $(t \in T)$

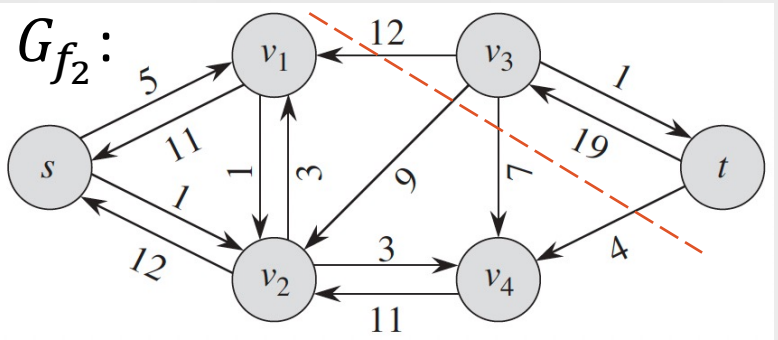
For any $u \in S$ and $v \in T$, we must have $c_f(u, v) = 0$.

By Lemma (on slide 11), we have $|f| = f(S, T)$.

Need to show: $f(S, T) = c(S, T)$.



G with f_2



Max-Flow Min-Cut Theorem (3/3)

Max-Flow Min-Cut Theorem If f is a flow in a flow network $G = (V, E)$ with source s and sink t, the the following conditions are equivalent:

2) The residual network G_f contains no augmenting paths.

3) $|f| = c(S, T)$ for some cut (S, T) of G.

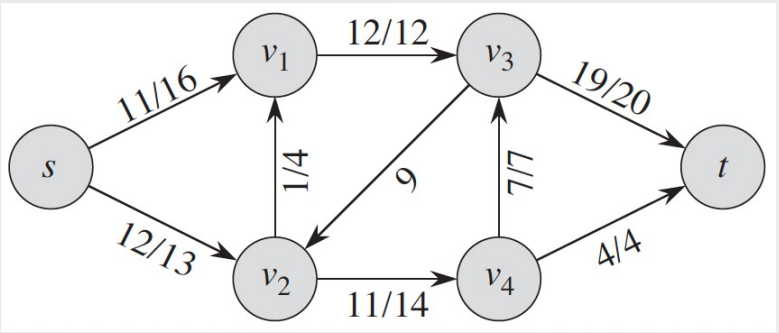
Proof. (cont'd) (2) ⇒ (3): Show that $f(S, T) = c(S, T)$.

For any $u \in S$ and $v \in T$, we must have $c_f(u, v) = 0$.

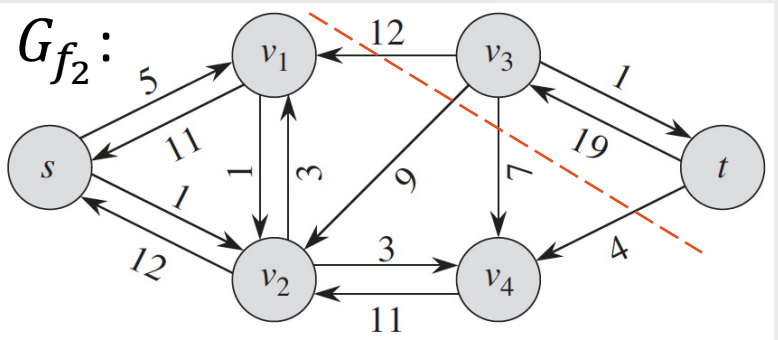
Based on definitions of residual capacity,

- If $(u, v) \in E$, then $c(u, v) = f(u, v)$
- If $(v, u) \in E$, then $f(v, u) = 0$

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u)$$
$$= \sum_{u \in S} \sum_{v \in T} c(u, v) = c(S, T)$$ □



G with f_2



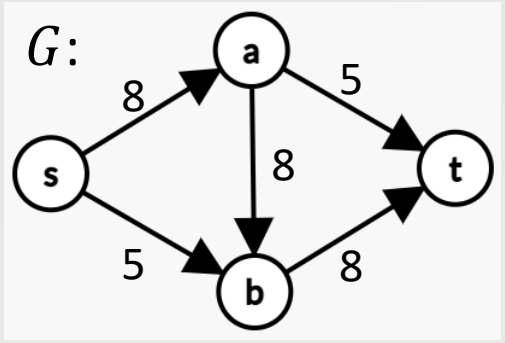
The Ford-Fulkerson Method

The Basic Ford-Fulkerson Algorithm

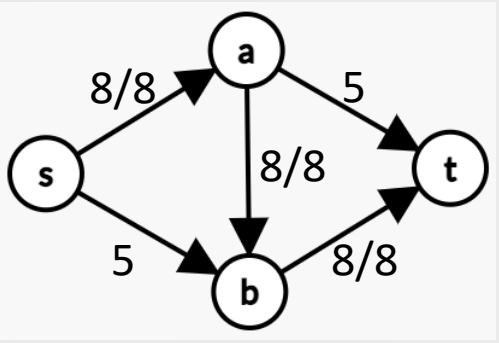
The Basic Ford-Fulkerson Algorithm

FORD-FULKERSON(G, s, t)
1 for each edge $(u, v) \in G.E$
2 $\quad (u, v).f = 0$
3 build the residual network G_f
4 **while** there exists a path p from s to t in G_f
5 $\quad$ augment f by $c_f(p)$
6 $\quad$ update G_f

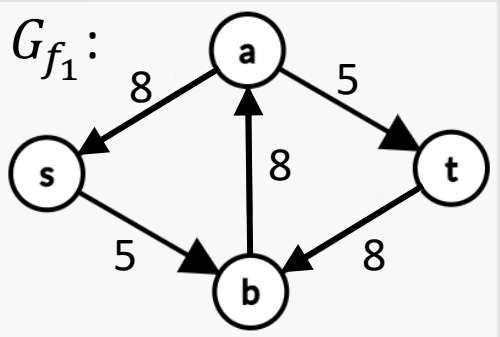
Example 1:



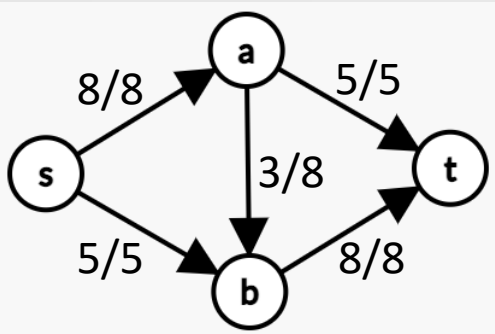
$c_{f_0}(p_1) = 8$



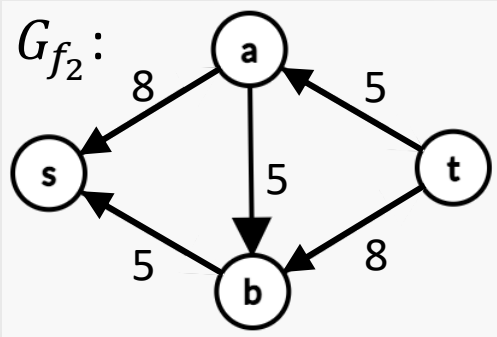
G with f_1



$c_{f_1}(p_2) = 5$



G with f_2



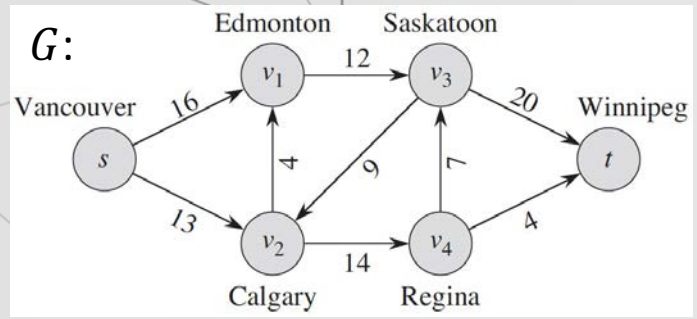
No augmenting path

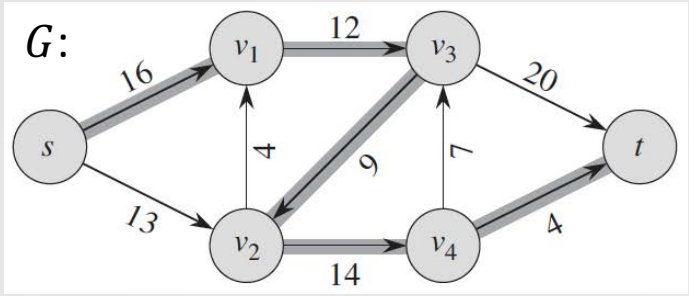
- f_2 is a max flow in G.
- $|f_2| = 13$
- A min-cut of G:

  $S = \{s\}$

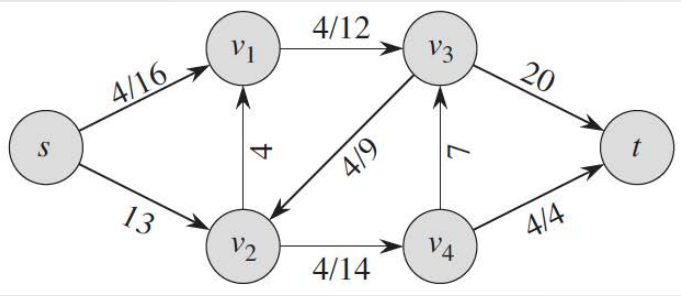
  $T = \{a, b, t\}$

The Trucking Problem Example (1/2)

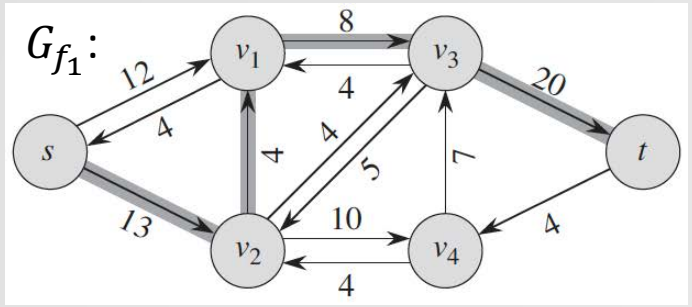




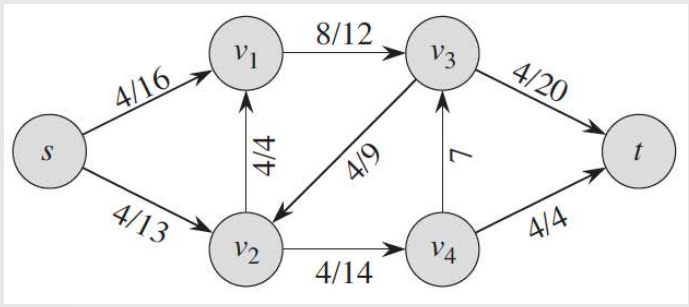
$c_{f_0}(p_1) = 4$



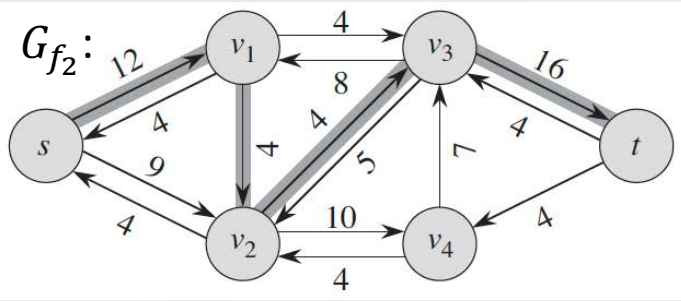
G with f_1



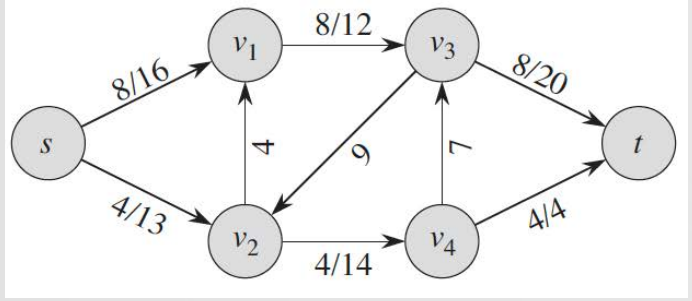
$c_{f_1}(p_2) = 4$



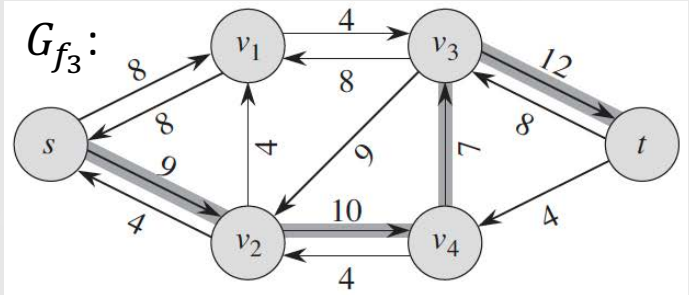
G with f_2



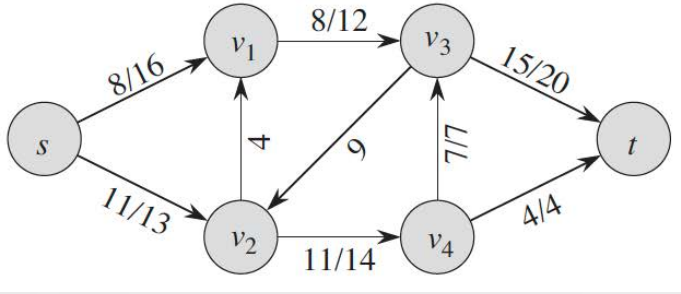
$c_{f_2}(p_3) = 4$



G with f_3

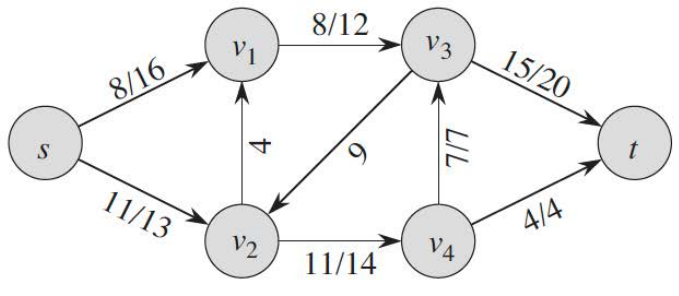


$c_{f_3}(p_4) = 7$

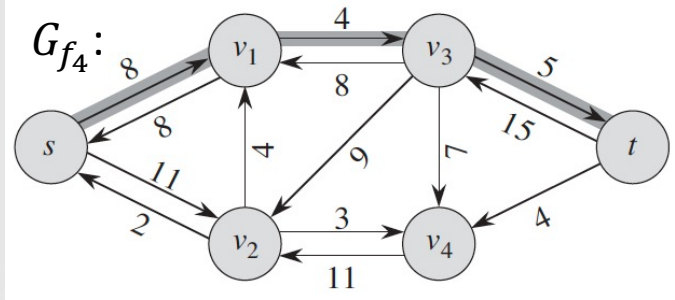


G with f_4

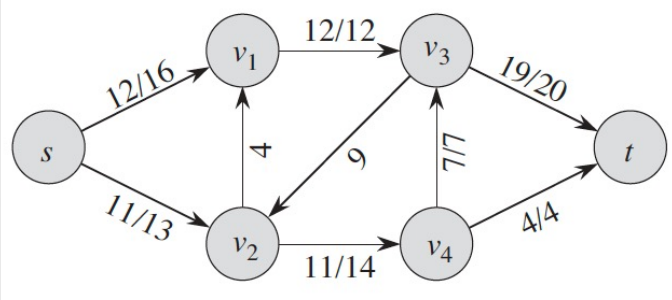
The Trucking Problem Example (2/2)



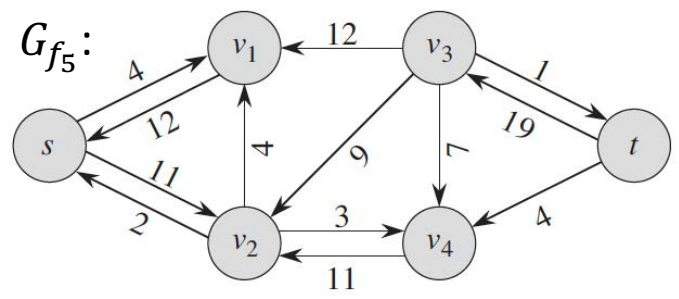
G with f_4



$c_{f_4}(p_5) = 4$



G with f_5



No more augmenting path.

- f_5 is a maximum flow in G.
- $|f_5| = 23$
- A min-cut of G:

  $S = \{s, v_1, v_2, v_4\}$

  $T = \{v_3, t\}$

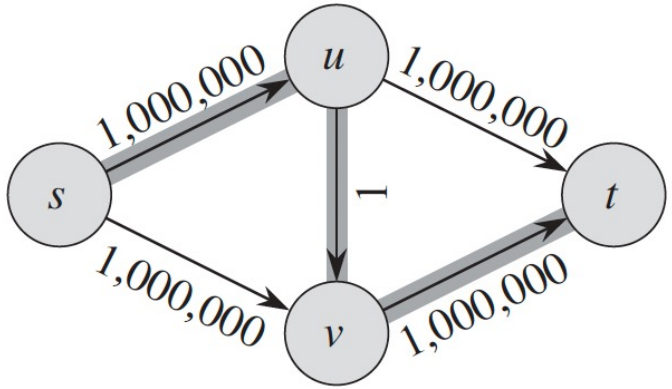
S contains all the vertices that can be reached from s in G_{f_5}.

Time Complexity of Ford-Fulkerson Algorithm

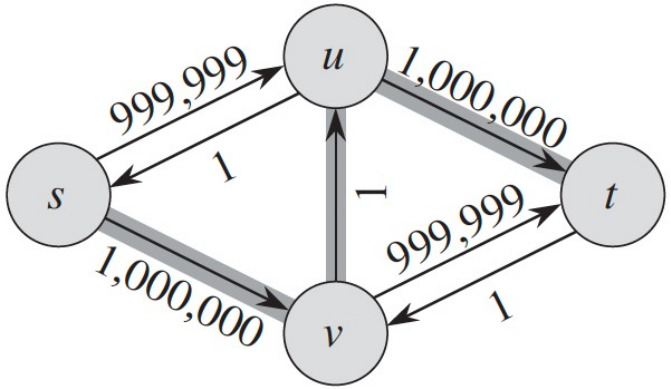
FORD-FULKERSON(G, s, t)
1 for each edge $(u, v) \in G.E$
2 $\quad (u, v).f = 0$
3 build the residual network G_f
4 **while** there exists a path p from s to t in G_f
5 $\quad$ augment f by $c_f(p)$
6 $\quad$ update G_f

- The time complexity depends on how we find the augmenting path in line 4.
- It can be very slow.
- **Example:**

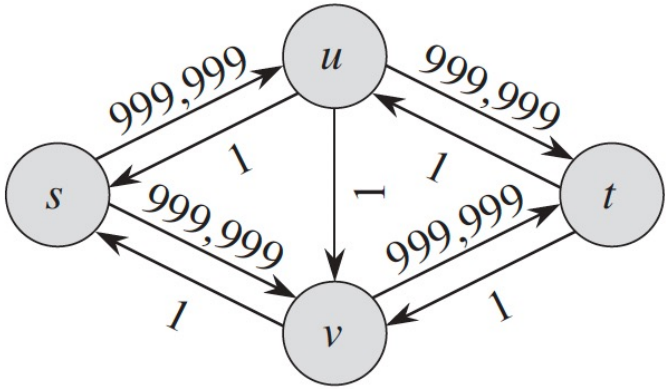
  It takes 2×10^6 iterations.



G



G_{f_1}



G_{f_2}

Analysis of Ford-Fulkerson Algorithm

```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ G.E
2      (u, v).f = 0
3  build the residual network G_f
4  while there exists a path p from s to t in G_f
5      augment f by c_f(p)
6      update G_f
```

- Lines 1-2: $O(m)$
- Line 3: $O(n+m)$
- Lines 4-6: in each iteration,
  - Line 4: $O(n+m)$ – by DFS or BFS
  - Line 5: $O(n)$
  - Line 6: $O(n)$

- **Assumption:** Capacities are all integers.
- Then, each augmenting path increases $|f|$ by at least 1. – Why?
- If max flow is f^*, then there will be at most $|f^*|$ iterations.
- Total running time: $O(m|f^*|)$ – NOT polynomial in input size!

The Ford-Fulkerson Method

The Edmonds-Karp algorithm

Edmonds-Karp Algorithm

- It follows the basic Ford-Fulkerson algorithm.
- Computes augmenting paths by using BFS in G_f to find the shortest path from s to t with all edge weights being 1. Call it the ***BFS path***.
- Time complexity of Edmonds-Karp algorithm: $O(nm^2)$
  - NO assumption on values of capacities
- That is, the number of iterations of the **while** loop is in $O(nm)$.*
  - In each iteration, G_f is updated with ≥ 1 edge deleted.
  - We will prove: Each edge in G_f can be deleted and reinserted back later for at most $n/2$ times.
  - Then, with $O(m)$ edges, there are $O(nm)$ iterations in total.

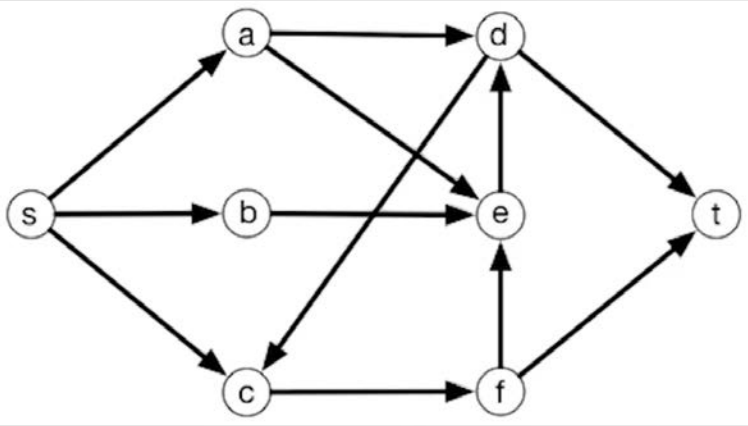
** This is Theorem 26.8 on p.729 of the textbook.*

Analysis of Edmonds-Karp Algorithm (1/5)

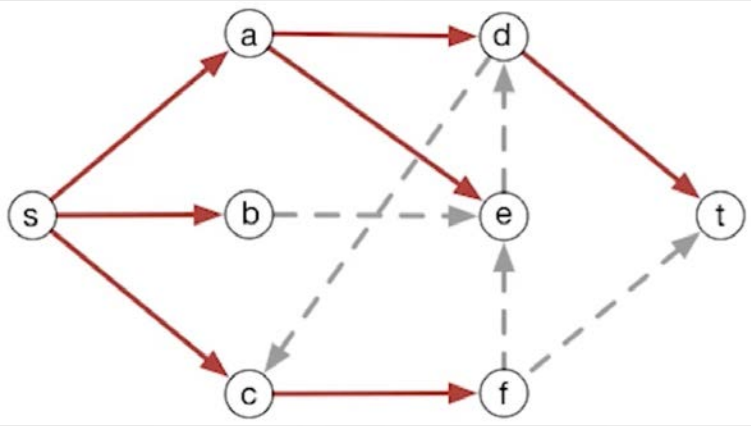
Lemma: Each edge (u, v) in G_f can be deleted and reinserted back later for at most $n/2$ times.

- *Recall:* BFS takes a graph G_f with starting vertex s. It can compute $v.d$ for every $v \in V$, where $v.d$ = distance (smallest # of edges) from s to v.
- For any edge (u, v) on a BFS path, $v.d = u.d + 1$.

Example:



G_f



Breadth-first tree of G_f

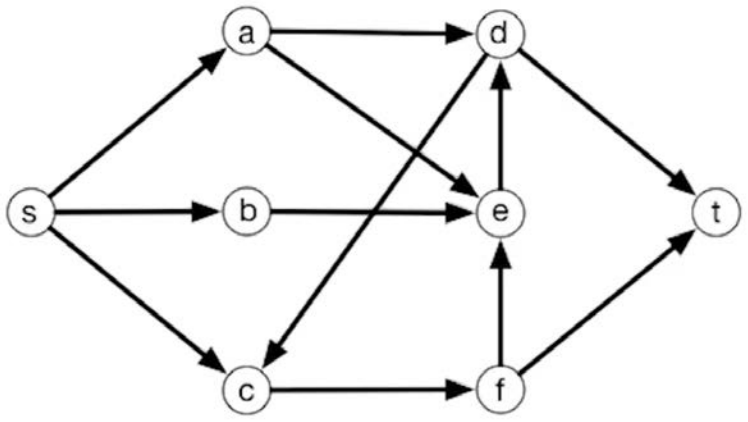
Analysis of Edmonds-Karp Algorithm (2/5)

Q: How does $v.d$ change as G_f changes?

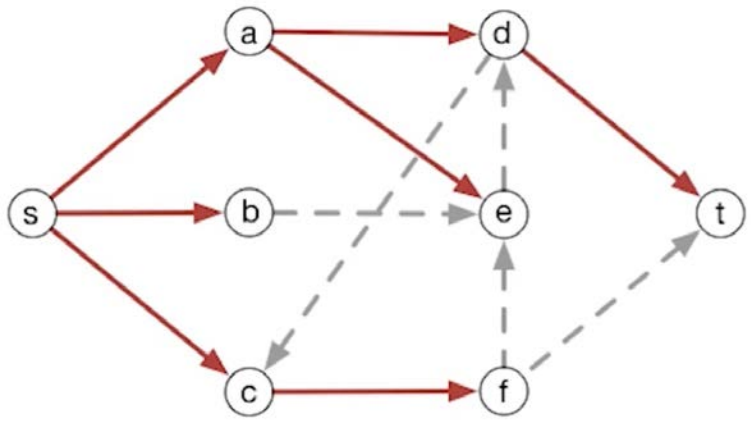
Claim: For every $v \in V$, $v.d$ never decreases.

Proof. When an edge in G_f is **deleted**, $v.d$ will not decrease. – Why?

- Only need to discuss the cases when an edge is **added** to G_f.
- **Q:** How does G_f change in each iteration?



G_f



Breadth-first tree of G_f

Analysis of Edmonds-Karp Algorithm (3/5)

Let f be the flow before the current iteration and f' be the flow updated with augmenting path (BFS path) p in the current iteration.

- *Recall:* Residual capacity is defined as:

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

For any edge (u, v),

- If (u, v) is **added** to G_f:
  - **Case 1:** $f(u,v) = c(u,v)$ and $f'(u,v) < c(u,v) \Rightarrow (v,u)$ is on p.
  - **Case 2:** $f(v,u) = 0$ and $f'(v,u) > 0 \Rightarrow (v,u)$ is on p.
- If (u, v) is **deleted** from G_f:
  - **Case 1:** $f(u,v) < c(u,v)$ and $f'(u,v) = c(u,v) \Rightarrow (u,v)$ is on p.
  - **Case 2**: $f'(v,u) = 0$ and $f(v,u) > 0 \Rightarrow (u,v)$ is on p.

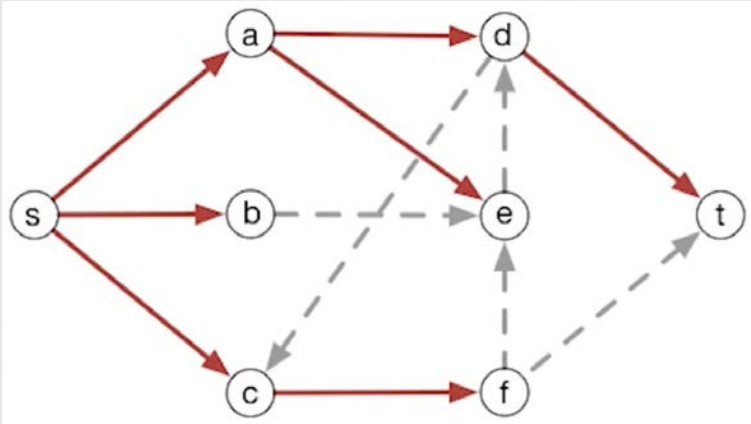
Analysis of Edmonds-Karp Algorithm (4/5)

Claim: For every $v \in V$, $v.d$ never decreases.

Proof. (cont'd) Discuss cases when an edge (u, v) is **added** to G_f.

We've shown: if an edge (u, v) is **added** to G_f, then (v, u) is on p.

- Suppose $v.d = k$ before edge (u, v) is **added** to G_f.
- As p is a BFS path, for (v, u), $u.d = v.d + 1 = k + 1$.
- So, adding (u, v) will not decrease $v.d$. – Why? □



Analysis of Edmonds-Karp Algorithm (5/5)

Lemma: Each edge (u, v) in G_f can be deleted and reinserted back later for at most $n/2$ times.

Proof. Suppose at some point, $u.d = k$.

- When edge (u, v) is **deleted** from G_f, (u, v) is on the BFS path p.
  So, $v.d = u.d + 1 \geq k + 1$.
- When edge (u, v) is later **added** back into $G_{f'}$, (v, u) is on the BFS path p'.
  So, $u.d' = v.d + 1 \geq k + 2$.

That is, $u.d$ will increase by at least 2.

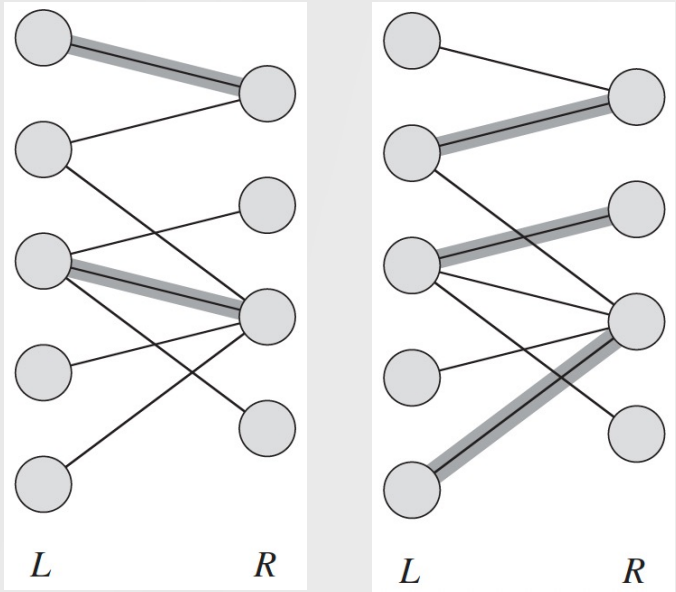
As $0 \leq u.d \leq n$, lemma is proved. □

Maximum Bipartite Matching

The Maximum-Bipartite-Matching Problem

- A graph $G = (V, E)$ is ***bipartite*** if we can partition $V = L \cup R$ $(L \cap R = \emptyset)$ such that all edges in E go between L and R.
- In an undirected graph $G = (V, E)$, a ***matching*** is a subset of edges $M \subseteq E$ s.t. for all $v \in V$, at most one edge of M is incident on v.
  - Vertex v is ***matched*** by M if an edge of M is incident on it; otherwise v is ***unmatched***.
  - A ***maximum matching*** is a matching of maximum cardinality, that is, a matching M s.t. $|M| \geq |M'|$ for any matching M'.
- ***The Maximum-Bipartite-Matching Problem***:
  - **Given:** A bipartite graph $G = (L \cup R, E)$
  - **Goal:** Find a maximum matching.

Example:

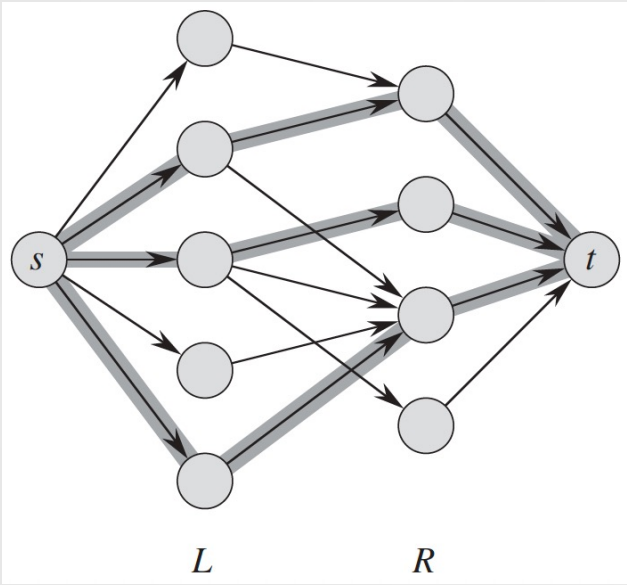


Maximum-Bipartite-Matching as Maximum-Flow

Given a bipartite graph $G = (L \cup R, E)$, we define the corresponding flow network $G' = (V', E')$ as follows:

- $V' = V \cup \{s, t\}$,
- $E' = \{(s, u) : u \in L\} \cup \{(u, v) : (u, v) \in E\} \cup \{(v, t) : v \cup R\}$,
- $c(u, v) = 1$ for all $(u, v) \in E'$.

Example:



Integrality Theorem

Integrality Theorem[*]. If the capacities of all edges in a flow network are integers, then

1. The value of the maximum flow f produced by the Ford-Fulkerson method, $|f|$, is also an integer.
2. For every edge (u, v), the value of $f(u, v)$ is an integer.

Lemma[**]: Let $G = (L \cup R, E)$ be a bipartite graph and $G' = (V', E')$ be its corresponding flow network. Then, a matching M in G corresponds to a flow f in G', with $|M| = |f|$.

Proof. See next slide.

---

* *This is Theorem 26.10 on p.734 of the textbook.*

** *See Lemma 26.9 on p.733-734 of the textbook for a complete proof.*

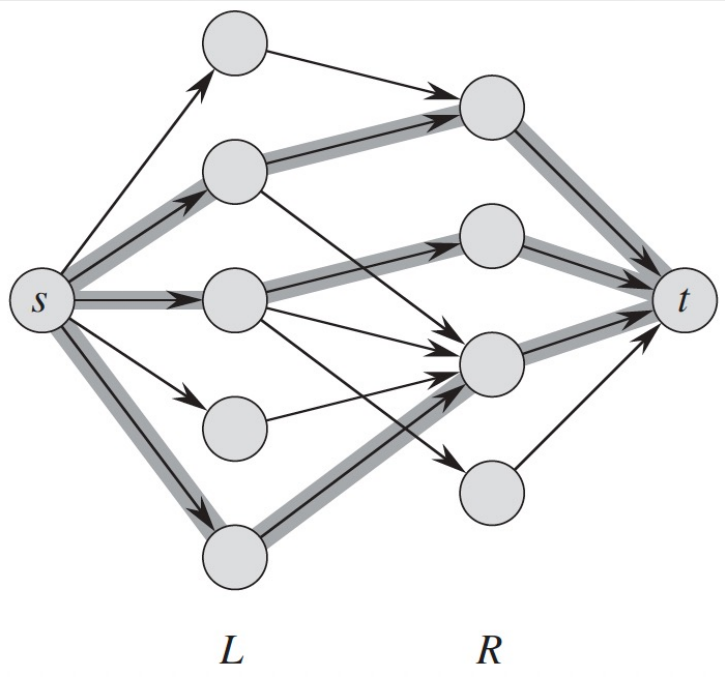
Correspondence Between Matchings and Flows (1/2)

Lemma: Let $G = (L \cup R, E)$ be a bipartite graph and $G' = (V', E')$ be its corresponding flow network. Then, a matching M in G corresponds to a flow f in G', with $|M| = |f|$.

Proof.

- "⇒": Show $|M| = k \Longrightarrow \exists f$ s.t. $|f| = k$.
  - For each edge $(u, v) \in M$, with $u \in L, v \in R$, $f(u, v) = 1, f(s, u) = 1$, and $f(v, t) = 1$.
  - $|f| = \sum_{v \in V} f(s, v) = k$.
- "⇐": Show $|f| = k \Longrightarrow \exists M$ s.t. $|M| = k$.
  - If $f(u, v) > 0$, then $f(u, v) = 1$
  - $M = \{(u, v) : u \in L, v \in R, f(u, v) = 1\}$
  - M is a matching. – Why?
  - $|M| = f(L \cup \{s\}, R \cup \{t\}) = |f| = k$. □

Example:



Correspondence Between Matchings and Flows (2/2)

Integrality Theorem. If the capacities of all edges in a flow network are integers, then

1. The value of the maximum flow f produced by the Ford-Fulkerson method, $|f|$, is also an integer.
2. For every edge (u, v), the value of $f(u, v)$ is an integer.

Lemma: Let $G = (L \cup R, E)$ be a bipartite graph and $G' = (V', E')$ be its corresponding flow network. Then, a matching M in G corresponds to a flow f in G', with $|M| = |f|$.

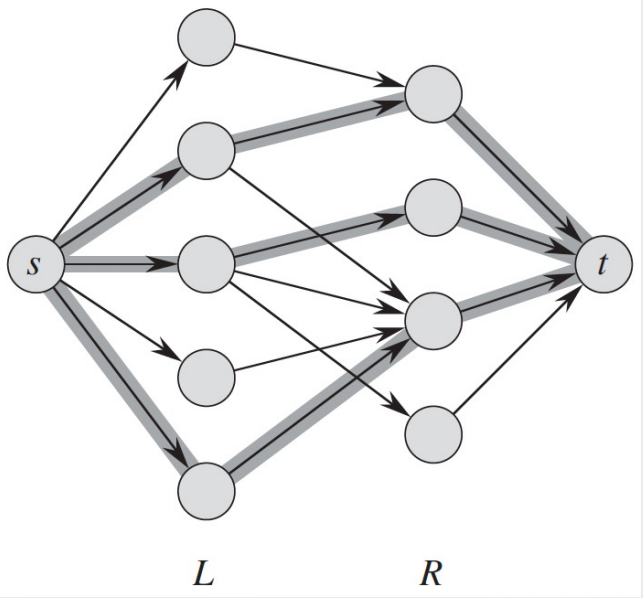
The **Integrality Theorem** and the **Lemma** lead to the following conclusion.

Corollary *(26.11)*: The cardinality of a maximum matching in a bipartite graph = the value of a maximum flow in its corresponding flow network.

Finding a Maximum Bipartite Matching

Given a bipartite graph $G = (L \cup R, E)$, with $n = |L \cup R|$ and $m = |E|$,

1. Create flow network $G' = (V', E')$
   - $|V'| = n + 2$ and $|E'| = m + n \in \Theta(m)$;
   - $c(u, v) = 1$ for all $(u, v) \in E'$.
   - Running time: $\Theta(m + n)$
2. Apply the Ford-Fulkerson method on G'
   - Let M be max matching in G,
   - $|M| \leq \min\{|L|, |R|\} \in O(n)$.
   - If is f' is max flow in G', then $|f'| = |M| \in O(n)$.
   - Running time: $O(mn)$



Thank you!
Questions?