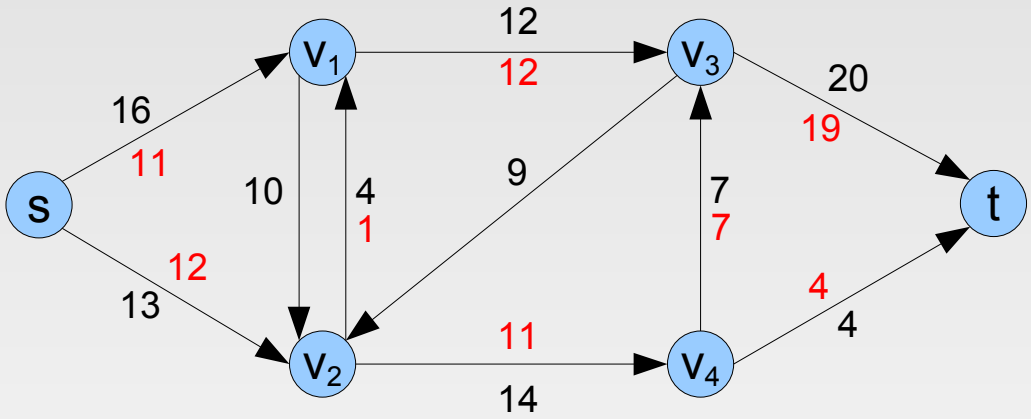
Maximum Flow & Bipartite Matching



Outline

- Introduce the **Maximum Flow** problem and potential applications.
- Formally define **Flow Networks** and the Maximum Flow problem.
- Study the **Ford-Fulkerson** method.
- Analyze the **Edmunds-Karp** algorithm.
- Solve the **Bipartite Matching** problem by reducing it to a Max Flow problem.

Flow Networks

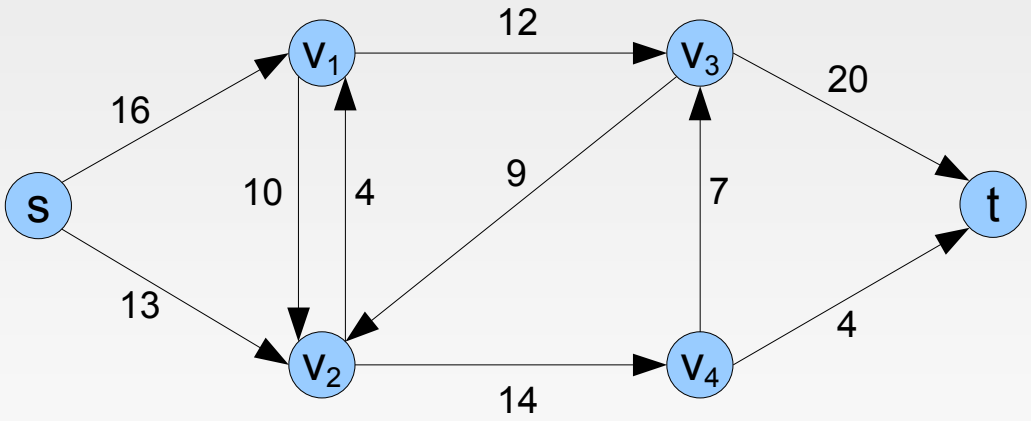
- Use a directed graph to model material that flows through conduits:
  - Each edge represents a conduit, and has a **capacity**, which is an upper bound on the **flow rate** = units/time.
  - Two special vertices:
    - the **source** *s* (where material is produced).
    - the **sink** *t* (where material is consumed).
    - the other vertices just distribute the material from incoming edges to adjacent vertices.

Flow Networks

- Examples of flow networks:
  - Network of pipes to transport fluid (oil, gas, water, etc):
    - edges are pipes, vertices are junctions of pipes.
  - Communication Networks:
    - edges are ”cables” of different bandwidth, vertices are routers.
- **Maximum Flow** = the maximum rate at which we can ship material from the **source** to the **sink**.

Flow Networks

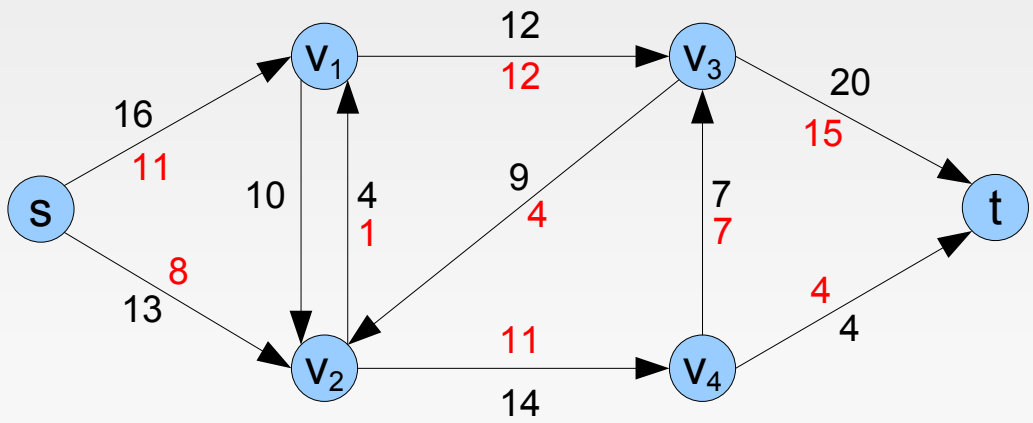
- **Flow Network** is a directed graph G=(V,E)
  - each edge (u,v) has a **capacity** c(u,v)≥0.
  - if (u,v)∉E, then assume c(u,v)=0.
  - two special vertices: **source** s and **sink** t.
  - assume s↦u↦t for any vertex u∈V.



Flow Networks

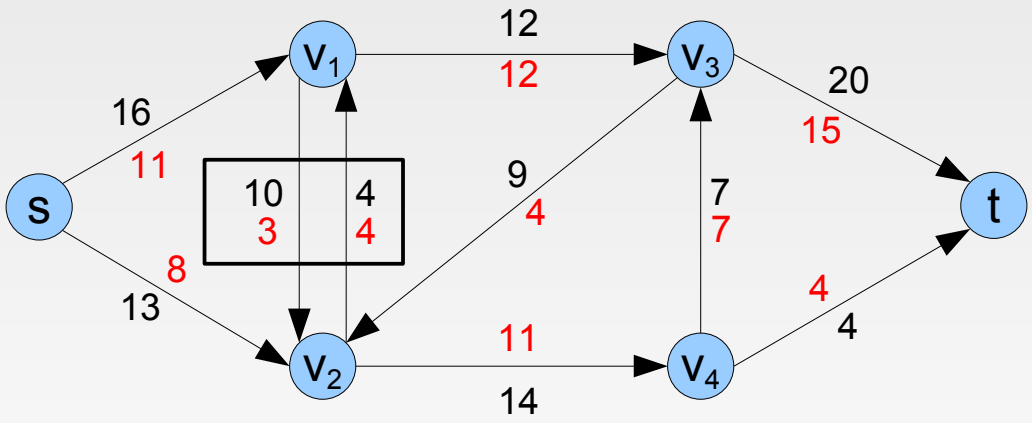
- **Flow** is a function f:VxV→R
  - **Capacity Constraint**: for all u,v∈V, $f(u,v) \leq c(u,v)$.
  - **Skew Symmetry**: for all u,v∈V, $f(u,v) = -f(v,u)$
  - **Flow Conservation**: for all u∈V−{s,t}

$$\sum_{v \in V} f(u,v) = f(u,V) = 0$$



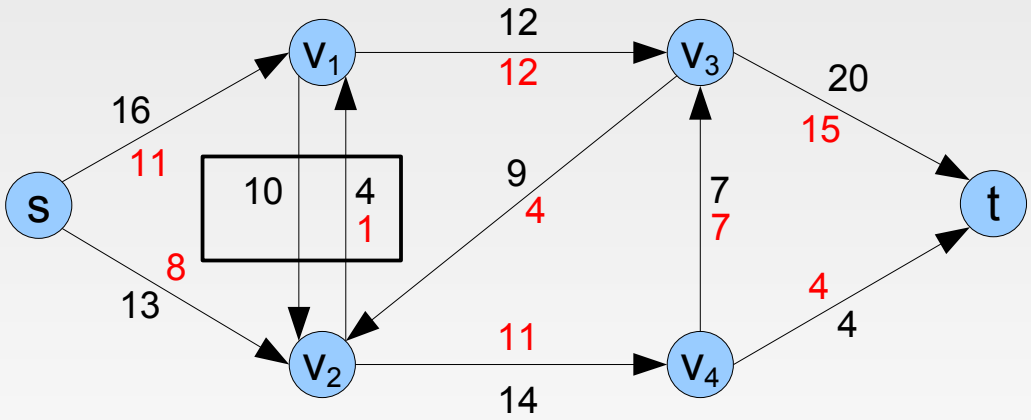
Flow Networks

- Conventions:
  - Only positive flows are shown.
  - Positive flows going both directions between two vertices are reduced to one positive flow.



Flow Networks

- Conventions:
  - Only positive flows are shown.
  - Positive flows going both directions between two vertices are reduced to one positive flow.

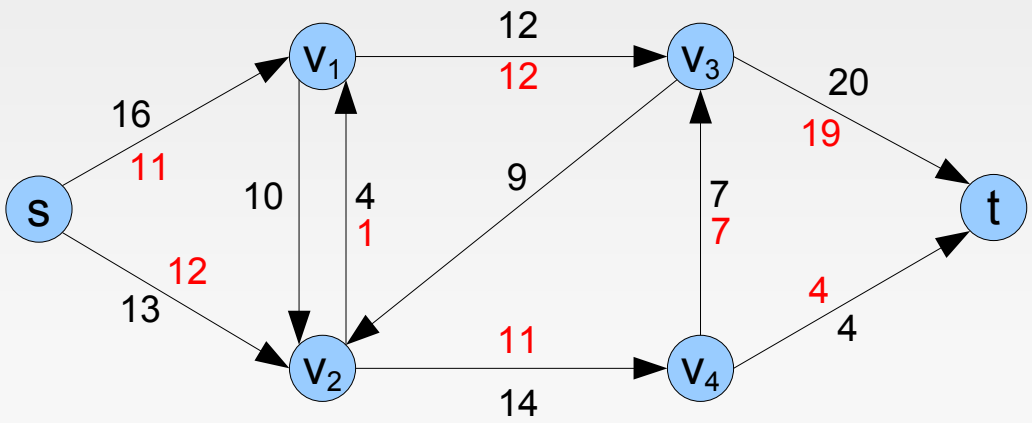


Maximum Flow

- The **value** of flow f in graph G:

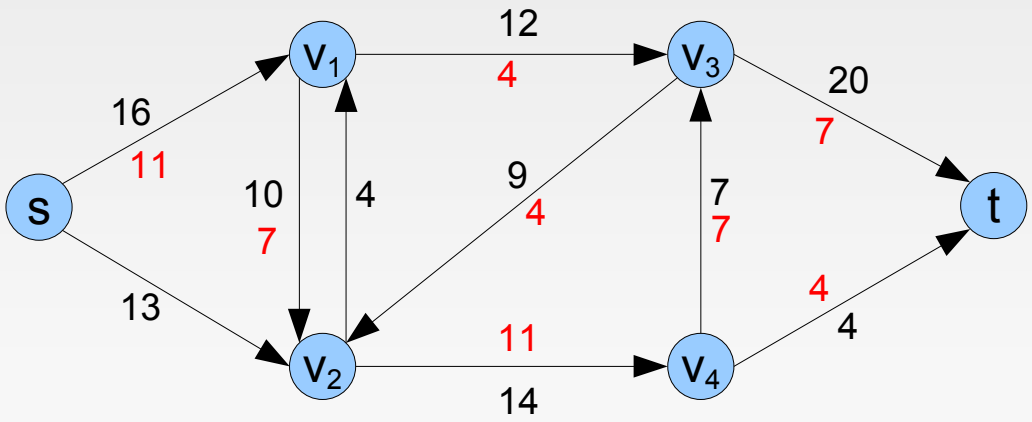
$$|f| = \sum_{v \in V} f(s,v) = f(s,V) = f(V,t)$$

- The **Maximum Flow** problem: given a flow network G with source s and sink t, find a flow f of maximum value |f|.



Basic Idea

- Given a flow f in a graph G, find a path p from s to t that can accept more flow.
  - **Augmenting Path**: a path p from s to t such that there is an a > 0 s.t. for all edges (u,v) along the path p we have f(u,v) + a ≤ c(u,v).
- Then increase the flow along p => better flow.



The Ford-Fulkerson method

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

1. initialize flow f to 0
2. **while** there exists an augmenting path p **do**
3. augment flow f along p
4. **return** f

Q_1: How do we find augmenting paths?

Q_2: How much can we augment the flow for each p?

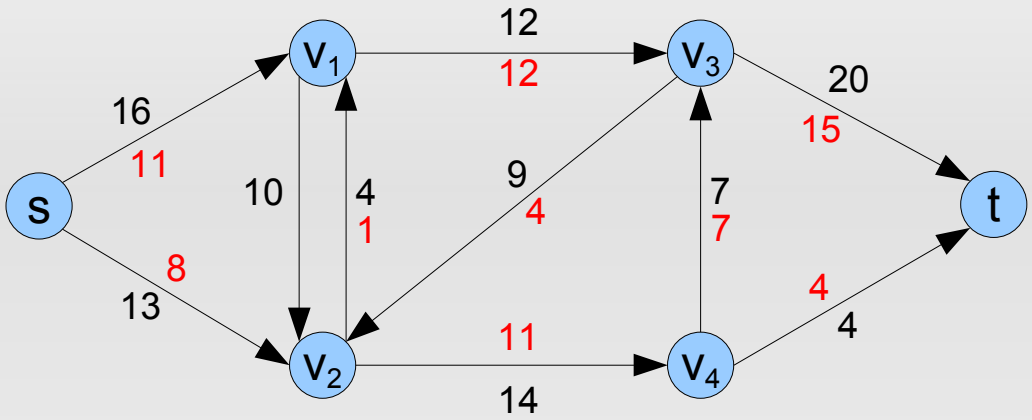
Q_3: Correctness & Time complexity.

Residual Network

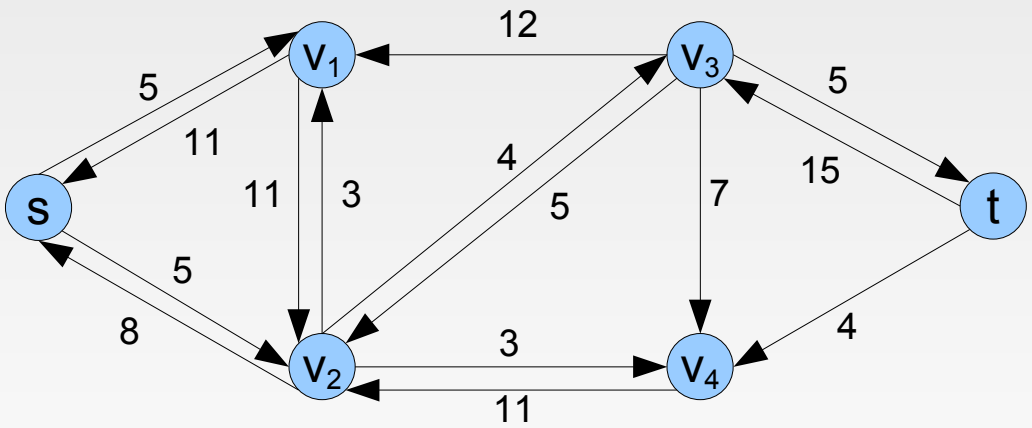
- Q_1: How do we find augmenting paths?
- A_1: Augmenting paths = paths between s and t in the **residual network** of G induced by f:
  - residual capacity $c_f(u,v)=c(u,v) - f(u,v)$
  - residual network $G_f=(V,E_f)$, $E_f=\{(u,v)\in VxV \mid c_f(u,v)>0\}$.
- *Observations:*
  - what happens when $f(u,v)<0$ and $c(u,v)=0$?
  - edges in E_f are edges in E or their reversals, hence $|E_f| \leq 2|E|$.

Residual Network: Example

$G=(V,E)$
$c \geq 0,\ f \leq c$

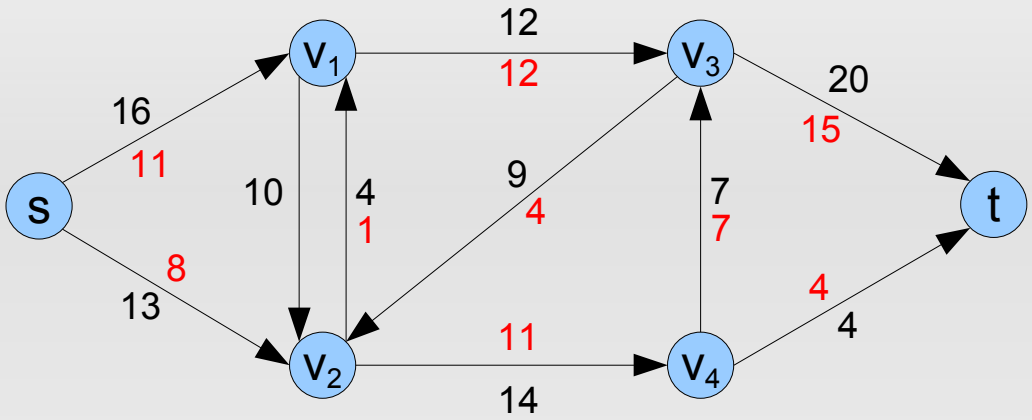


$G_f=(V,E_f)$
$c_f \geq 0$



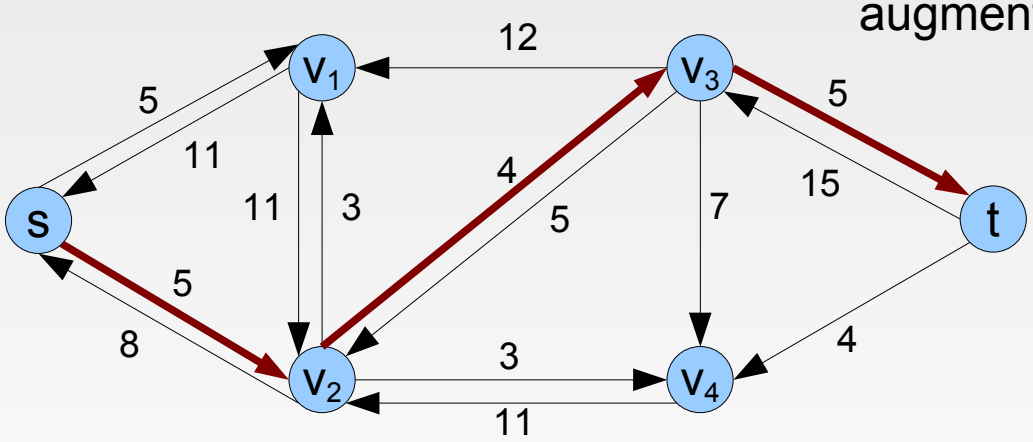
Residual Network: Example

G=(V,E)
$c \geq 0$, $f \leq c$



augmenting path in G_f

$G_f=(V,E_f)$
$c_f \geq 0$



Residual Capacity of a path

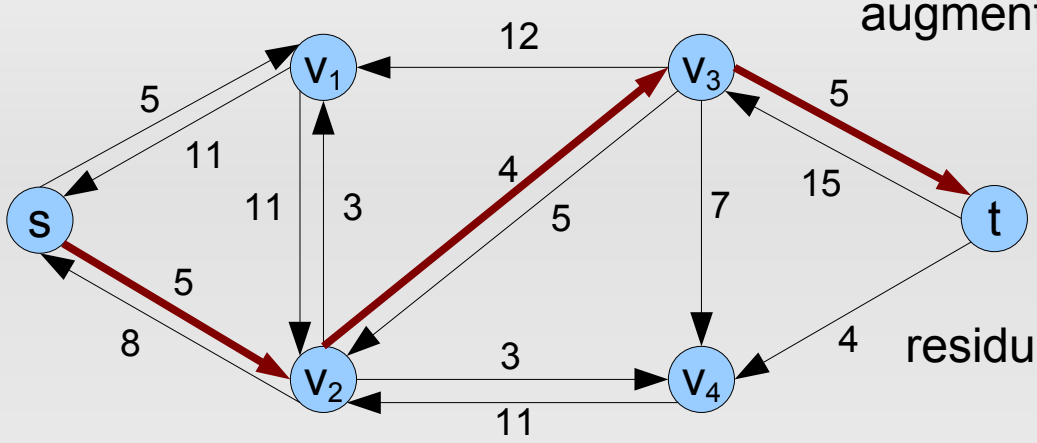
- Q_2: How much additional flow can we send along an augmenting path p?
- A_2: Compute the **residual capacity** of p:
  - $c_f(p)=\min\{c_f(u,v)|(u,v) \text{ is on } p\}$.
  - augment flow f along edges $(u,v) \in E$ as follows:
    - if (u,v) is on p, then $f'(u,v) = f(u,v) + c_f(p)$
    - if (v,u) is on p, then $f'(u,v) = f(u,v) – c_f(p)$
    - otherwise, $f'(u,v) = f(u,v)$.

Corollary 26.4

f' is a flow in G with value $|f'| > |f|$.

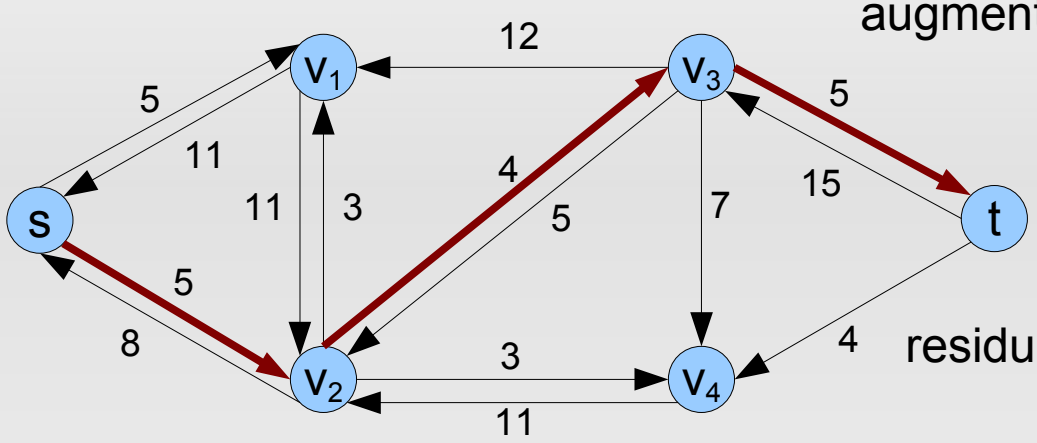
Residual Capacity & Augmentation

$G_f=(V,E_f)$
$c_f \geq 0$

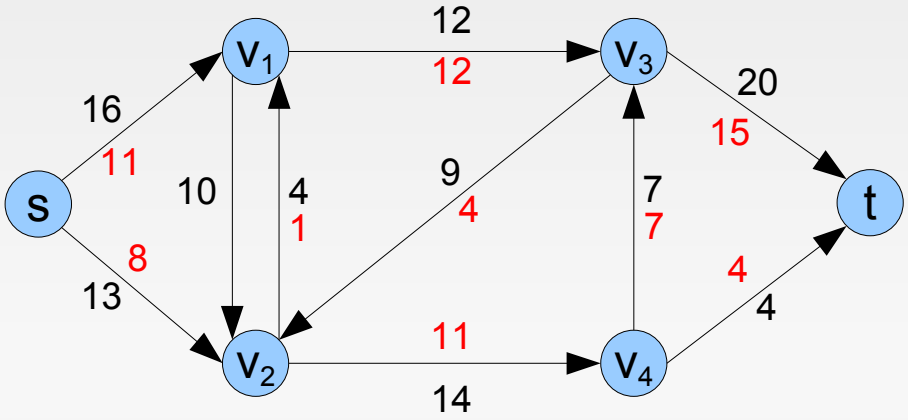


Residual Capacity & Augmentation

$G_f=(V,E_f)$
$c_f \geq 0$

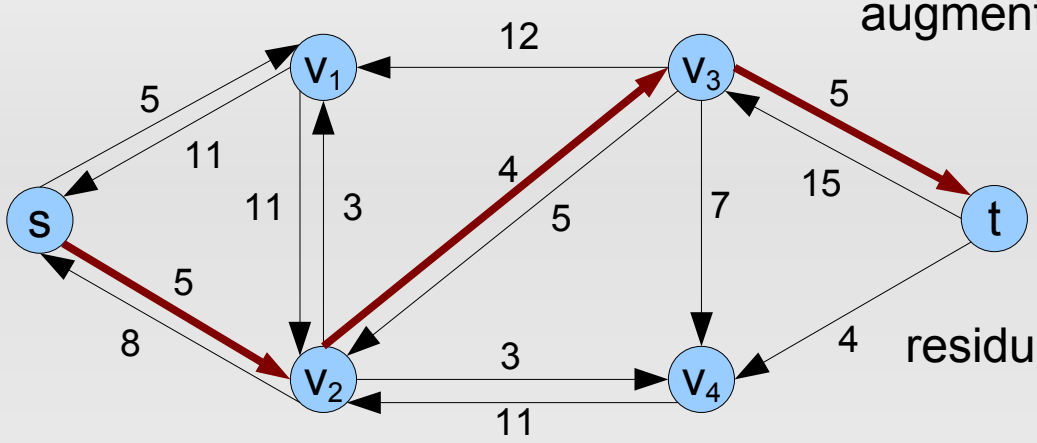


$G=(V,E)$
$c \geq 0, f \leq c$



Residual Capacity & Augmentation

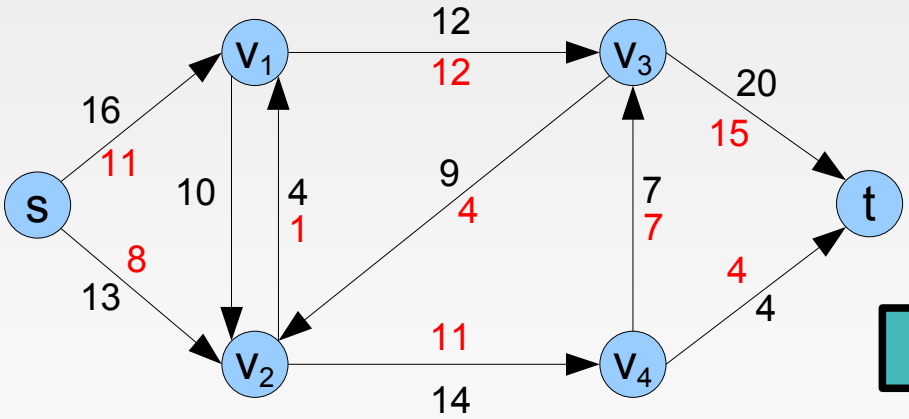
$G_f=(V,E_f)$
$c_f \geq 0$



augmenting path p in G_f

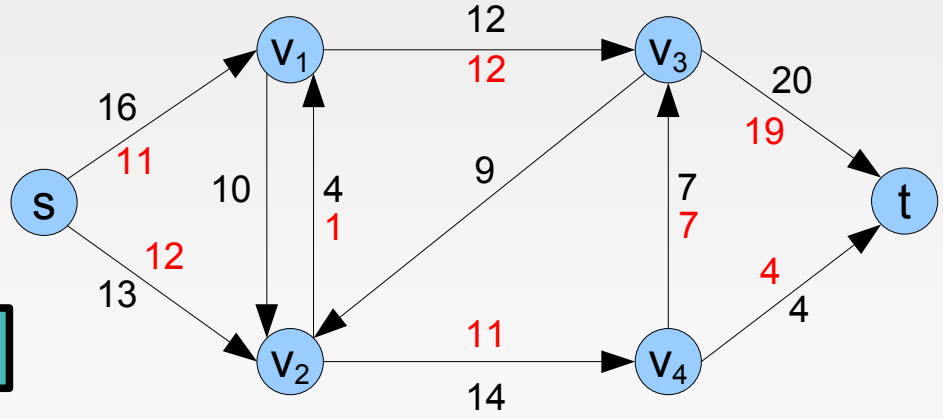
residual capacity $c_f(p)$ = ?

$G=(V,E)$
$c \geq 0, f \leq c$



|f| < |f'|

$G=(V,E)$
$c \geq 0, f' \leq c$



The Ford-Fulkerson Algorithm

Ford-Fulkerson(G,s,t)

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

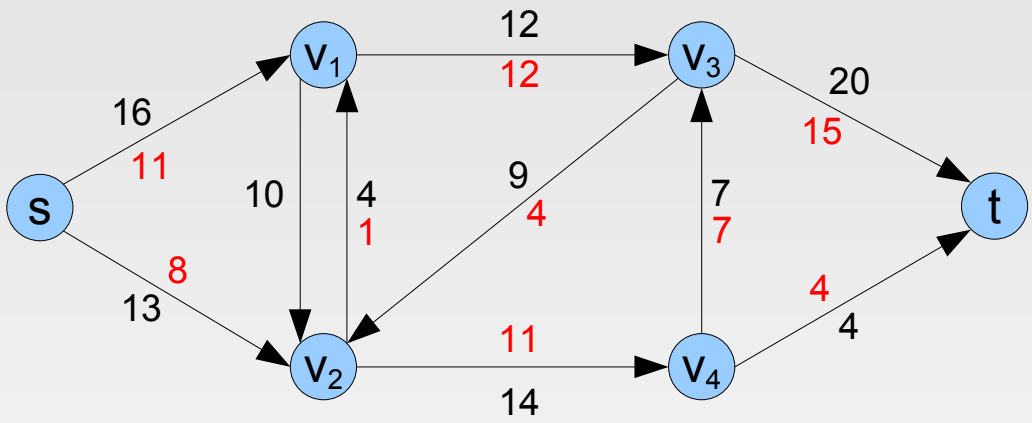
Ford-Fulkerson:Correctness

- To prove that the algorithm returns the max flow, we first define the **cut of a flow network**:
  - A cut of G = (V,E) is a partition of V into S and T=V–S such that s∈S and t∈T.
  - The net flow across the cut is: $f(S,T) = \sum_{u \in S, v \in T} f(u,v)$
  - The capacity of the cut is: $c(S,T) = \sum_{u \in S, v \in T} c(u,v)$
  - A **minimum cut** is a cut whose capacity is minimum over all cuts of the network.

Minimum Cut: Example

- What is the minimum cut in this flow network?

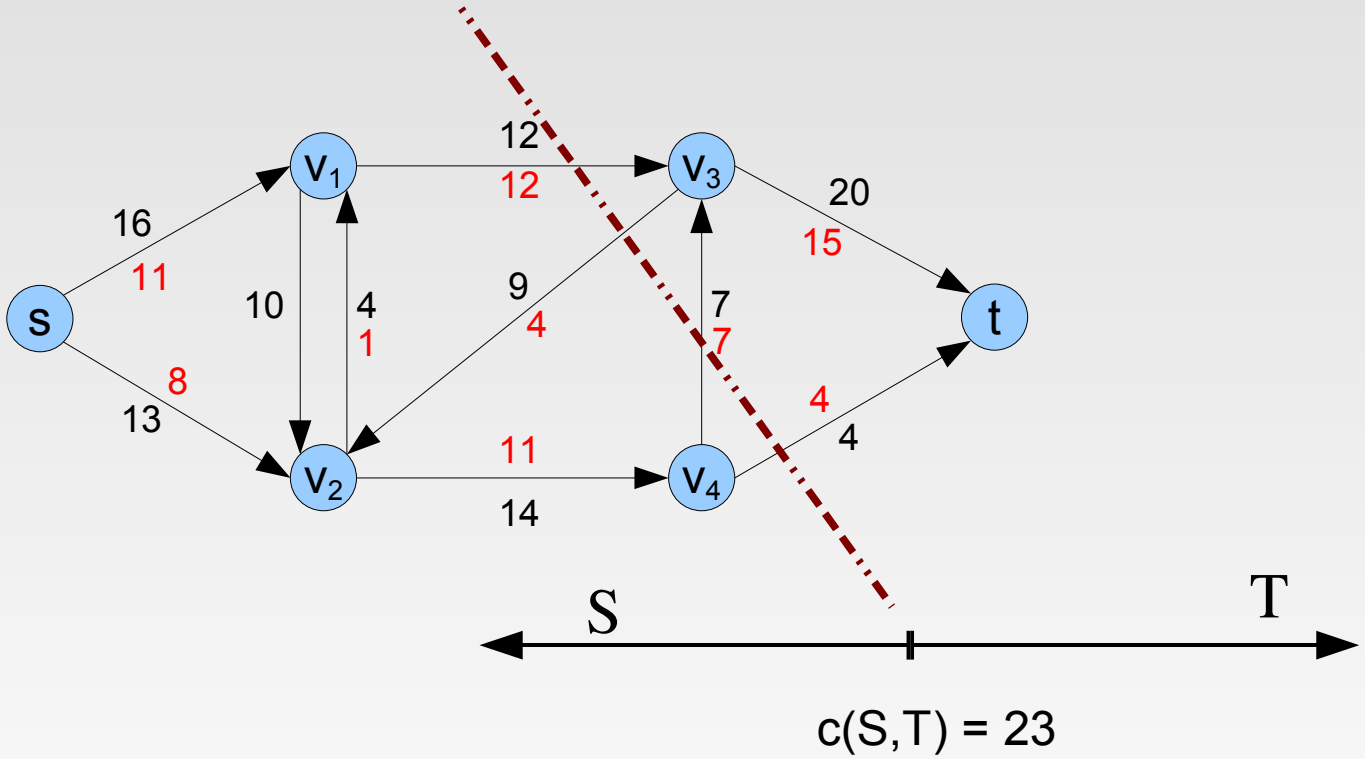
G=(V,E)
$c \geq 0, f \leq c$



Minimum Cut: Example

- What is the minimum cut in this flow network?

G=(V,E)
$c \geq 0$, $f \leq c$



Flows & Cuts

- Lemma 26.5
  - Let f be a flow in flow network G, and let (S,T) be a cut of G. Then the net flow across (S,T) is f(S,T) = |f|.
- Corollary 26.6
  - The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Hence, the maximum flow in a network is bounded from above by the capacity of a minimum cut of the network.

Max-flow Min-cut Theorem

- *Theorem 26.7*
  - If f is a flow in the flow network G, then the following conditions are equivalent:
    1. f is a maximal 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: $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$

The Ford-Fulkerson Algorithm

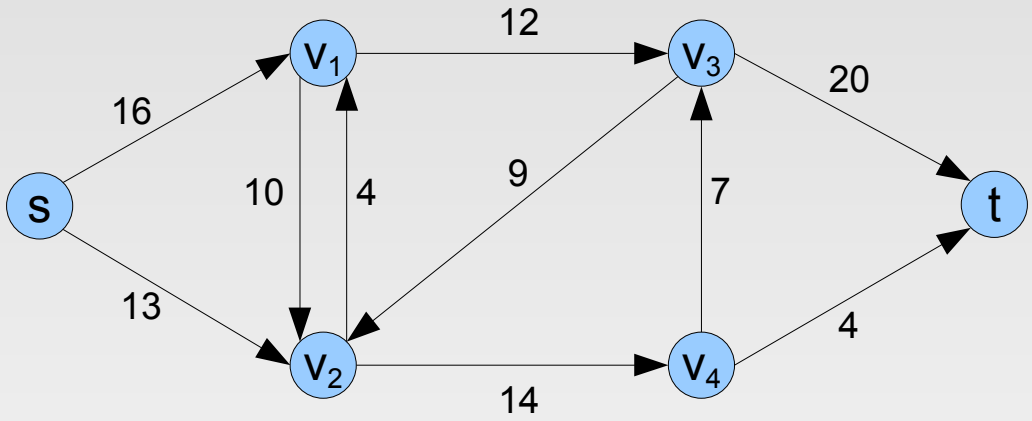
Ford-Fulkerson(G,s,t)

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

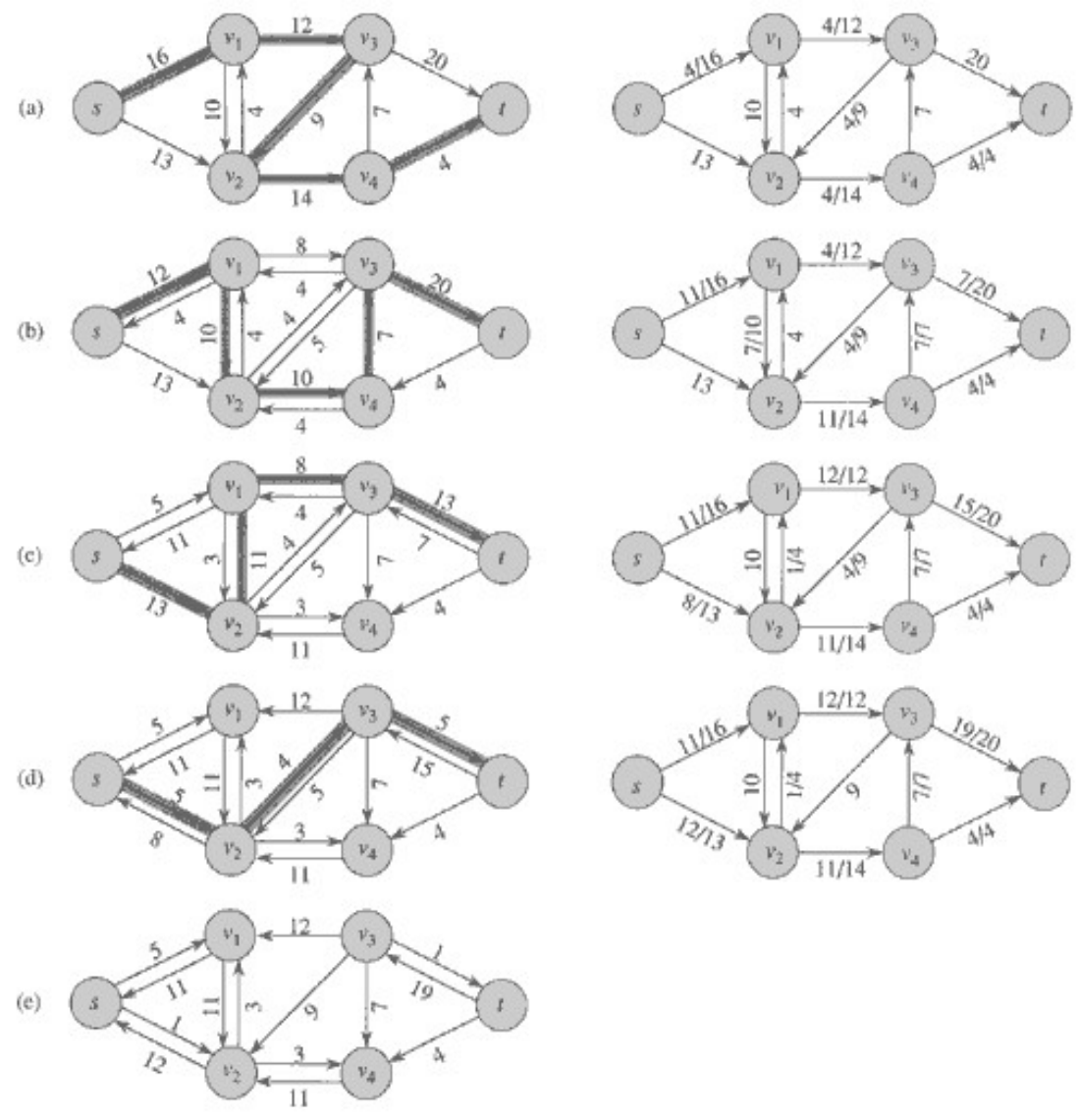
Ford-Fulkerson:Correctness

- The algorithm stops when no augmenting paths exist in the residual network G_f (step 3).
- By the Max-flow Min-cut theorem ($2 \Leftrightarrow 1$), this implies that f is a maximal flow in G.

Ford-Fulkerson: Example



Ford-Fulkerson: Example



Worst-case time complexity

- Time analysis – assume the capacities c(u,v) are **integral** numbers:
  - Let f* be the maximal flow returned by Ford-Fulkerson, and |f*| the max flow value.
  - Because f is increasing in each loop (steps 3 to 7), the loop cannot be executed more than |f*| times.
  - Use either depth-first or breadth-first search to find a path p in G_f, hence O(V+E) = O(E) time to find a path in step 3.
  - Loop executed at most |f*| times, hence O(|f*|E).

Worst-case time complexity

Ford-Fulkerson(G,s,t)

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

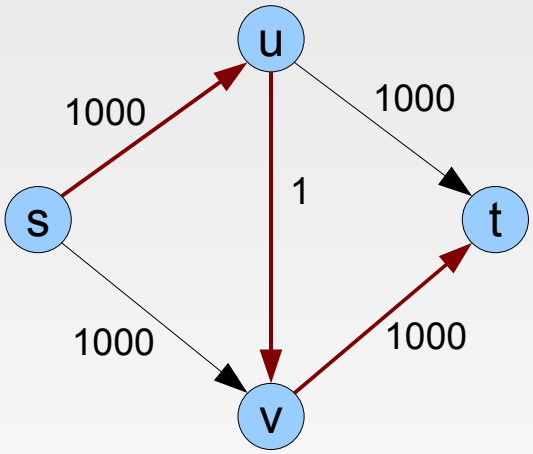
:O(|f*|E)

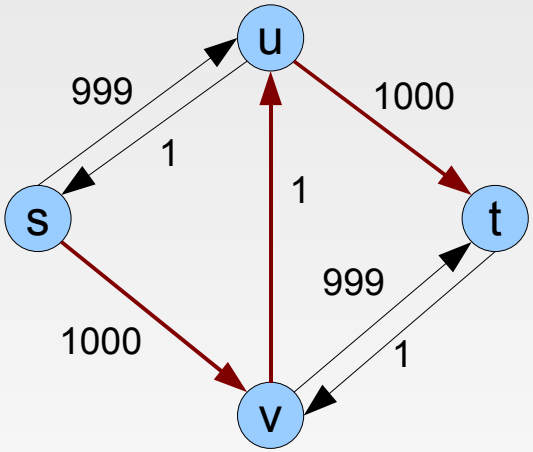
8. **return** f

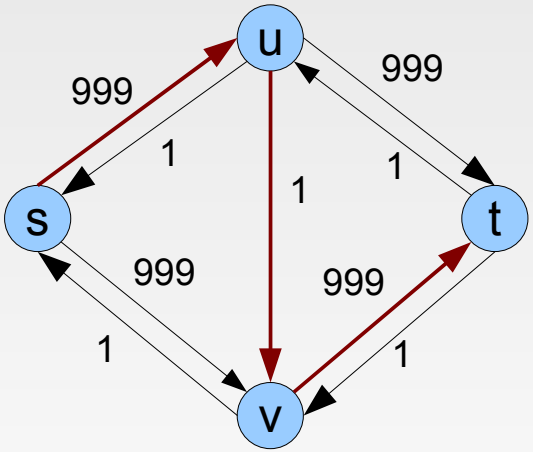
Use either depth-first or breadth-first search to find a path p in G_f, hence O(V+E) = O(E) time to find a path. Executed at most |f*| times, hence O(|f*|E).

Worst Case: Example

- A worst case that takes $\Theta(|f^*|E)$ time?
  - At every step, residual capacity is 1.
  - Max flow value is $|f^*|=2x1000$.

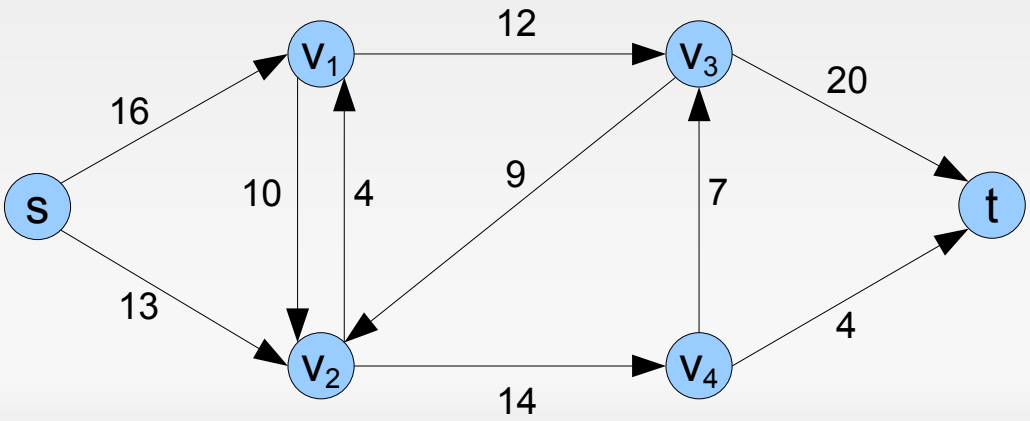






Finding augmenting paths

- Breadth First Search (22.2) vs. Depth FS (22.3):
  - Assume $s=v_0$ and $t=v_5$, adjacency lists have vertices in order of indeces, and each edge has unit length.
  - Find a path between between s and t:
    - DFS(s,t) = {s, ..., t} *intermediate vertices?*
    - BFS(s,t) = {s, ..., t} *intermediate vertices?*



Edmonds-Karp algorithm

- Take **shortest path** as an augmenting path:
  - Use **breadth-first search** to find shortest path from s to t in the residual network, in time O(E+V)=O(E).
  - Prove that # of augmentations is O(VE), hence overall time complexity is $O(VE^2)$:
    - *Definition*: an edge (u,v) on an augmenting path p is **critical** if it has the minimum residual capacity in the path: $c_f(u,v)=c_f(p)$
    - Prove that each edge becomes critical at most ~V/2 times *(next slide)*.
    - There are O(E) edges in the residual network, and each augmenting path has at least one critical edge ⇒ O(VE) augmenting paths.

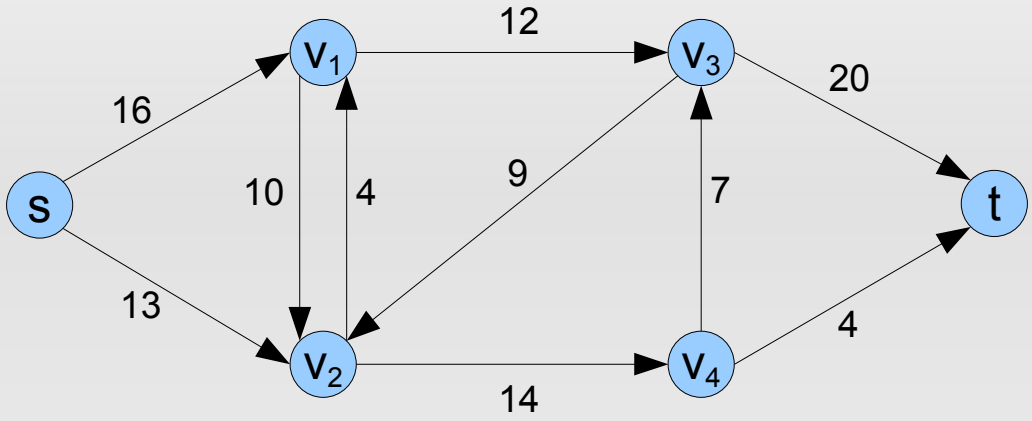
Edmonds-Karp algorithm

- An edge can become critical at most ~V/2 times:
  - edge (u,v) on an augmenting path is critical;
  - after augmentation, (u,v) dissapears from the network;
  - (u,v) reappears later only if the flow from u to v is decreased, which occurs only if (v,u) appears on an augmenting path;
  - *one can show that in-between these events, the distance between s and u has increased by at least 2.*
  - this can happen at most V/2 times.

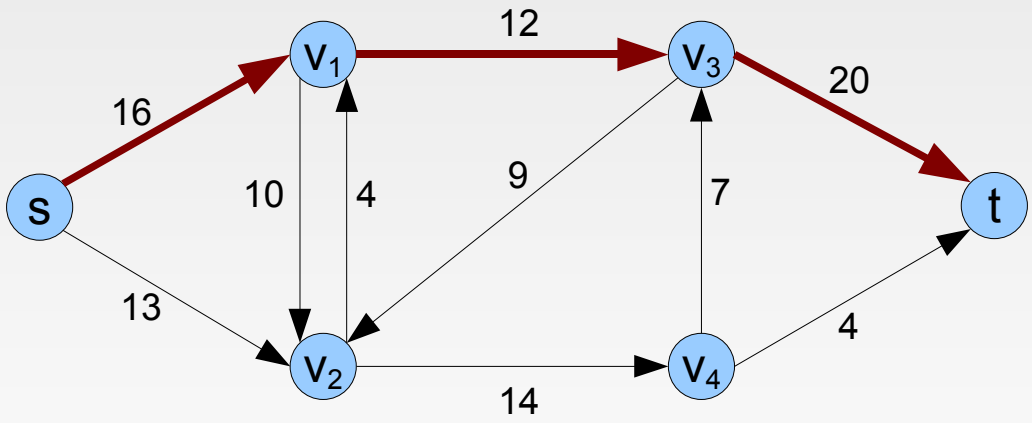
- Proved $O(VE^2)$ time complexity for Edmonds-Karp.

Edmonds-Karp: Example

Flow network

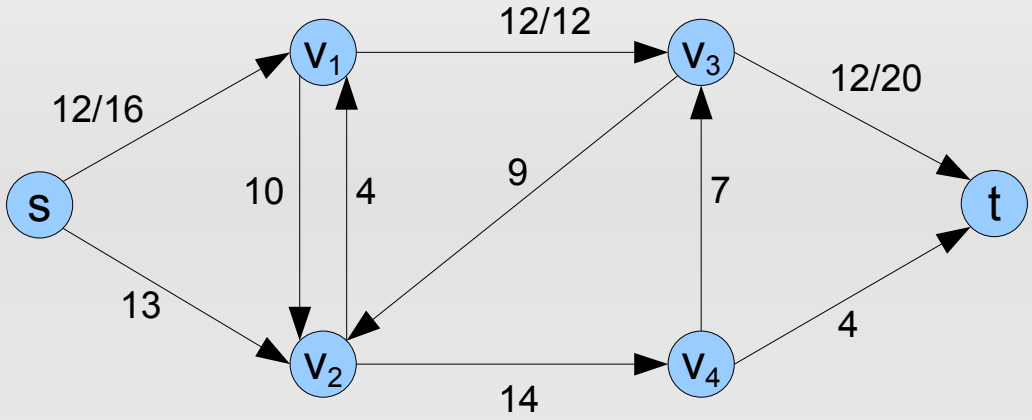


Residual network

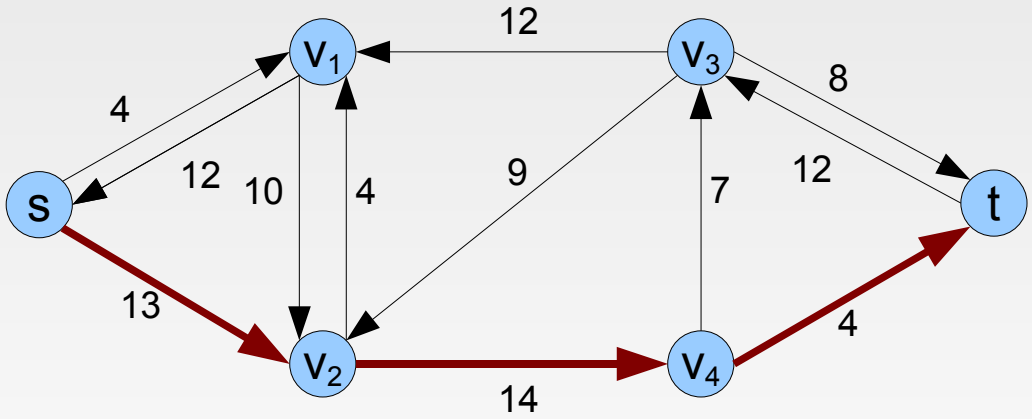


Edmonds-Karp: Example

Flow network

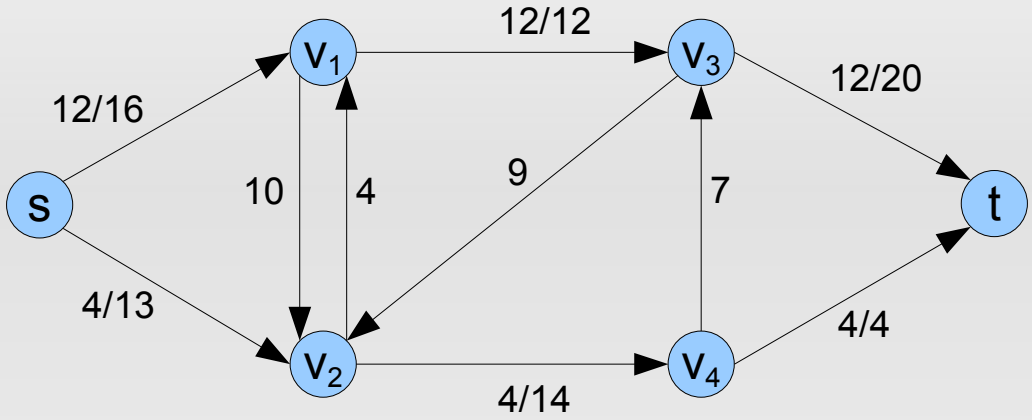


Residual network

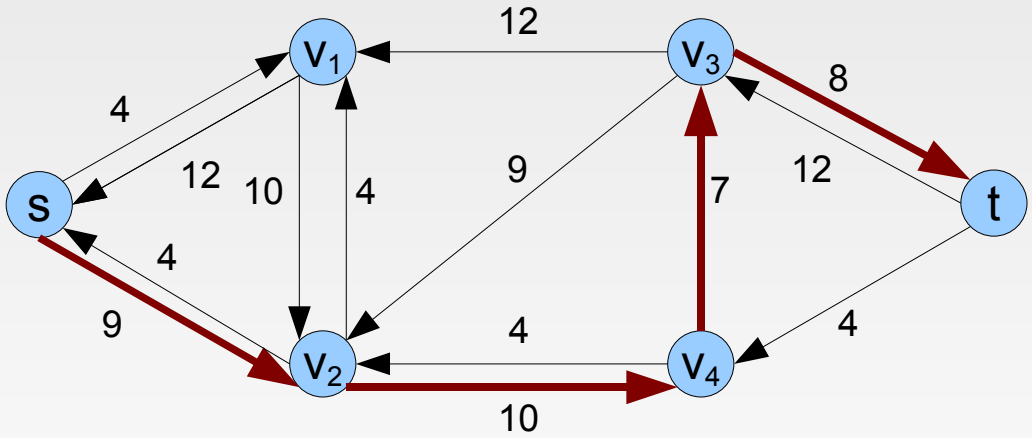


Edmonds-Karp: Example

Flow network

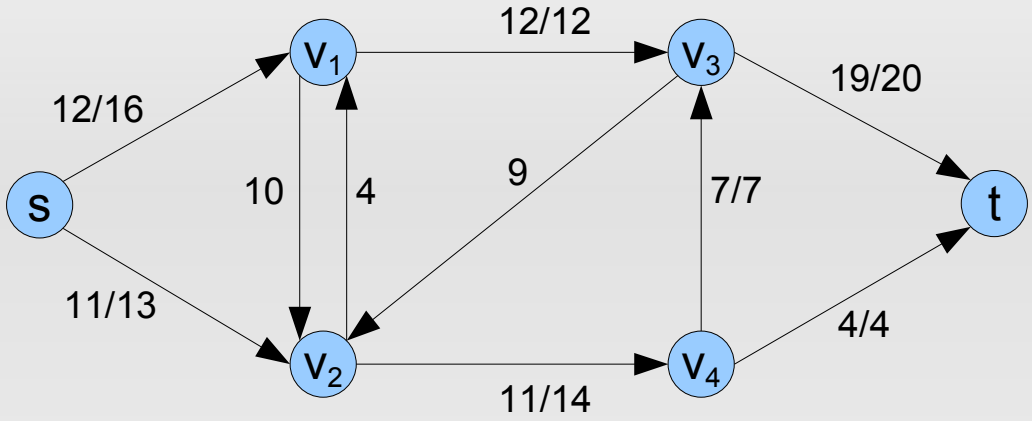


Residual network

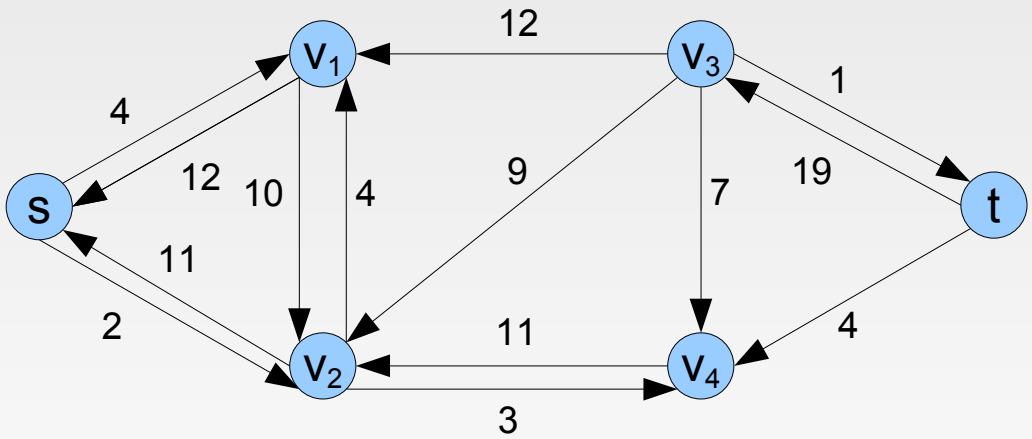


Edmonds-Karp: Example

Flow network



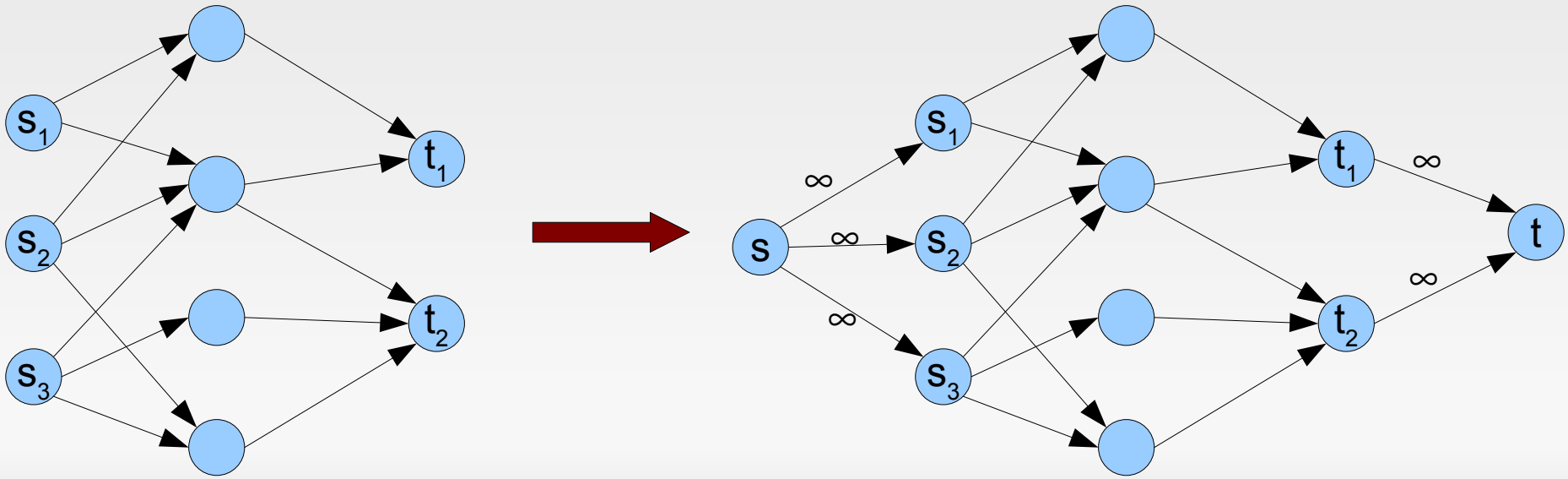
Residual network



No path between s and t ⇒ STOP!

Maximum Flow: Applications

1) Networks with multiple sources and sinks – reduce it to an ordinary max-flow problem:
   - Create *supersource* s, connect to sources $\{s_1, ..., s_m\}$.
   - Create *supersink* t, connect to sinks $\{t_1, ..., t_n\}$.
   - Set $c(s,s_i)=\infty$ and $c(t,t_j)=\infty$.

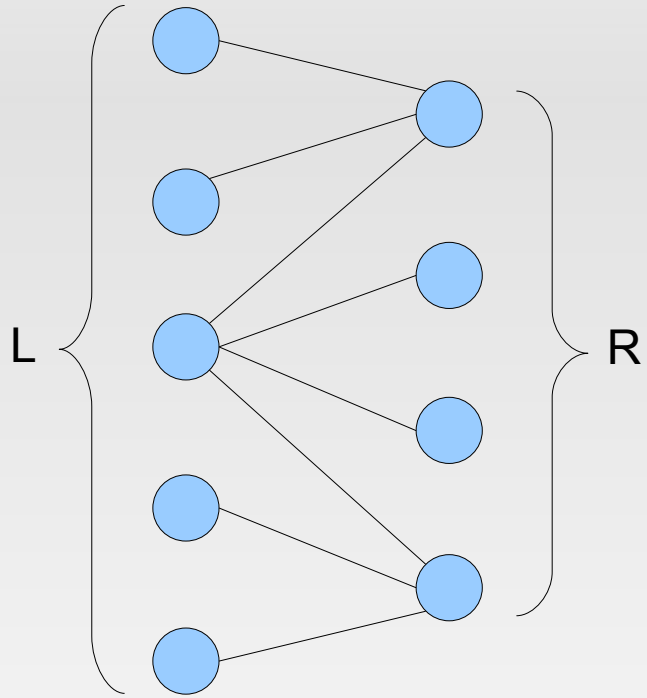


Maximum Flow: Applications

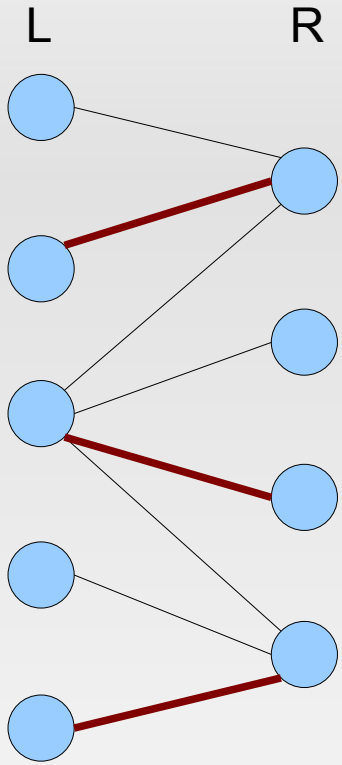
2) Maximum Bipartite Matching

- Undirected G=(V,E) is a **bipartite graph** if:
  - V can be partitioned into V = L∪R.
  - All edges in E go between L and R.
  - Assume every vertex has at least one incident edge.
- A a set of edges M⊆E is a **matching** in G=(V,E) if:
  - for all v∈E at most one edge of M is incident on v.
- The optimization problem:
  - Given a bipartite graph G=(L∪R, E), find a matching of maximum cardinality.

Maximum Bipartite Matching



A bipartite graph G=(L∪R,E).



A bipartite matching – is it maximum?

Maximum Bipartite Matching : Applications

1) Matching a set of machines L with a set R of tasks to be performed simultaneously:
   - An edge between machine $u \in L$ and task $v \in R$ indicates that machine u can execute task v.
   - Want to maximize the number of tasks executed.
2) Dating agency:
   - L = women, R = men
   - An edge indicate potential ”compatibility”.
   - Want to do as many matches as possible.

Max Matching as Max Flow

- Create a flow network G'=(V',E') in which flows correspond to matchings in original graph G=(V,E):
  - Add source s and sink t: $V' = V \cup \{s,t\}$.
  - E' is made of *directed* edges as follows:
    - from source s to vertices in L: $\{(s,u) | u \in L\}$
    - from u in L to v in R: $\{(u,v) | u \in L, v \in R, \text{ and } (u,v) \in E\}$
    - from vertices in R to sink t: $\{(v,t) | v \in R\}$
  - Assign unit capacity to each edge in E':
    - $c(u,v) = 1$, for all $(u,v) \in E'$

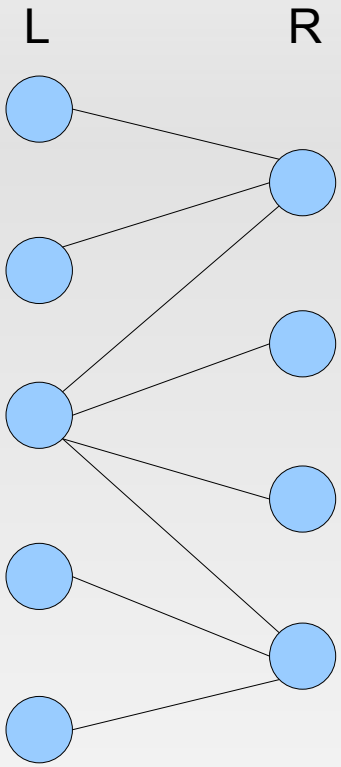
Max Matching as Max Flow

- Create a flow network G'=(V',E') in which flows correspond to matchings in original graph G=(V,E):
  - Add source s and sink t: $V' = V \cup \{s,t\}$.
  - E' is made of *directed* edges as follows:
    - from source s to vertices in L: $\{(s,u) | u \in L\}$
    - from u in L to v in R: $\{(u,v) | u \in L, v \in R, \text{ and } (u,v) \in E\}$
    - from vertices in R to sink t: $\{(v,t) | v \in R\}$
  - Assign unit capacity to each edge in E':
    - $c(u,v) = 1$, for all $(u,v) \in E'$

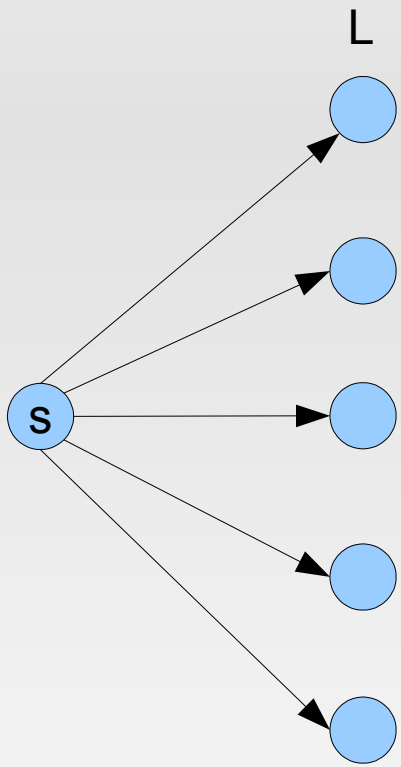
Corollary 26.12

Let M be a maximum matching M in bipartite graph G.
Let f be a maximum flow in flow network G'.
Then $|M| = |f|$.

Max Matching as Max Flow

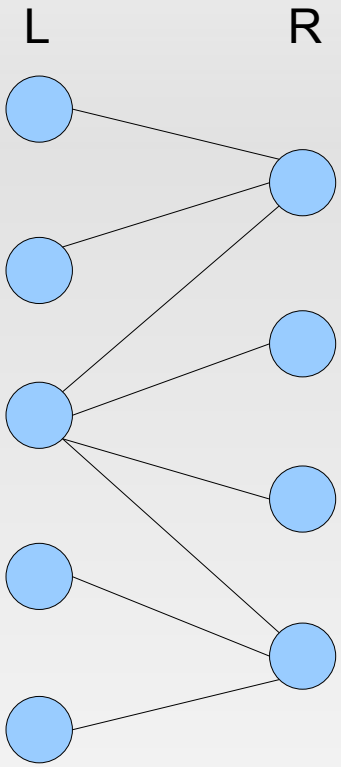


Bipartite Graph

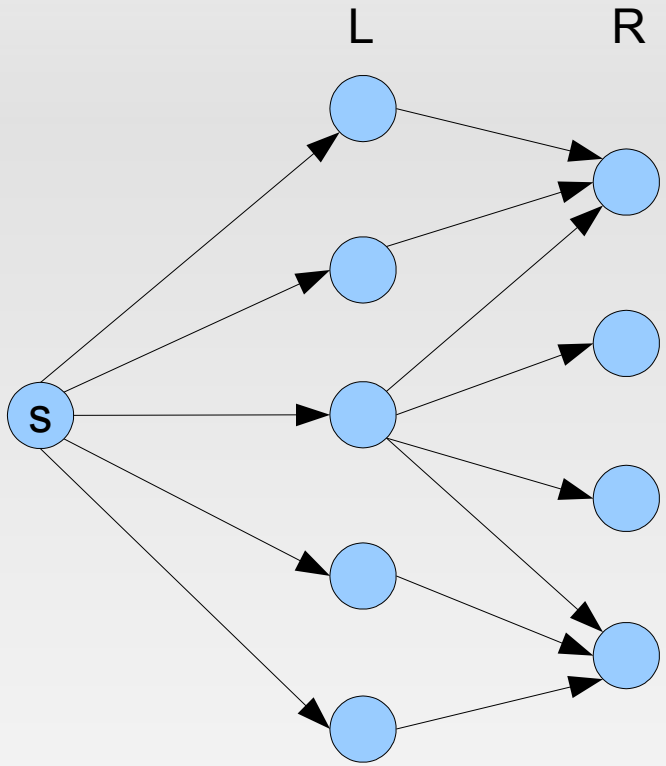


Flow Network

Max Matching as Max Flow

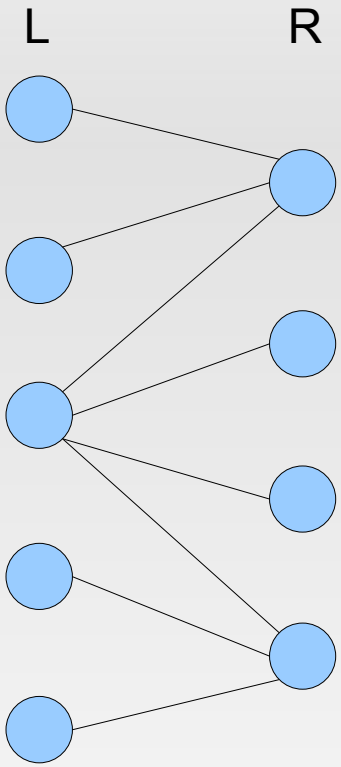


Bipartite Graph

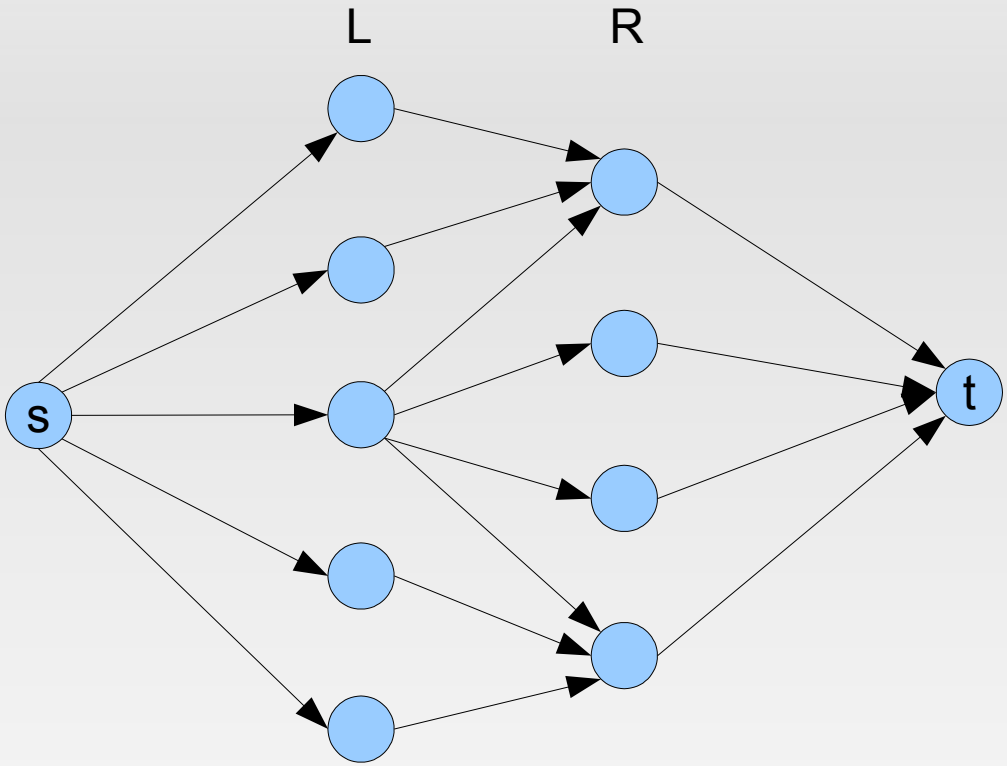


Flow Network

Max Matching as Max Flow

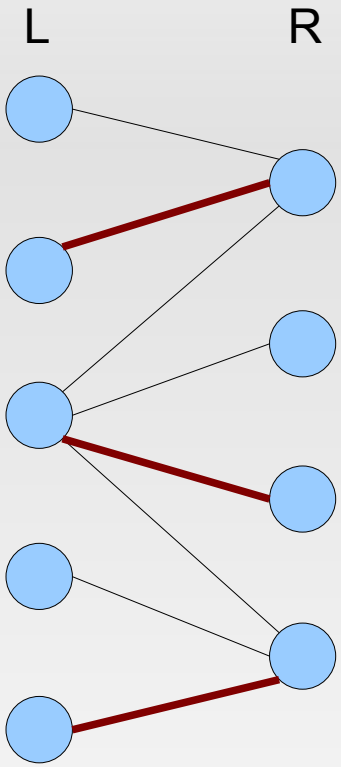


Bipartite Graph

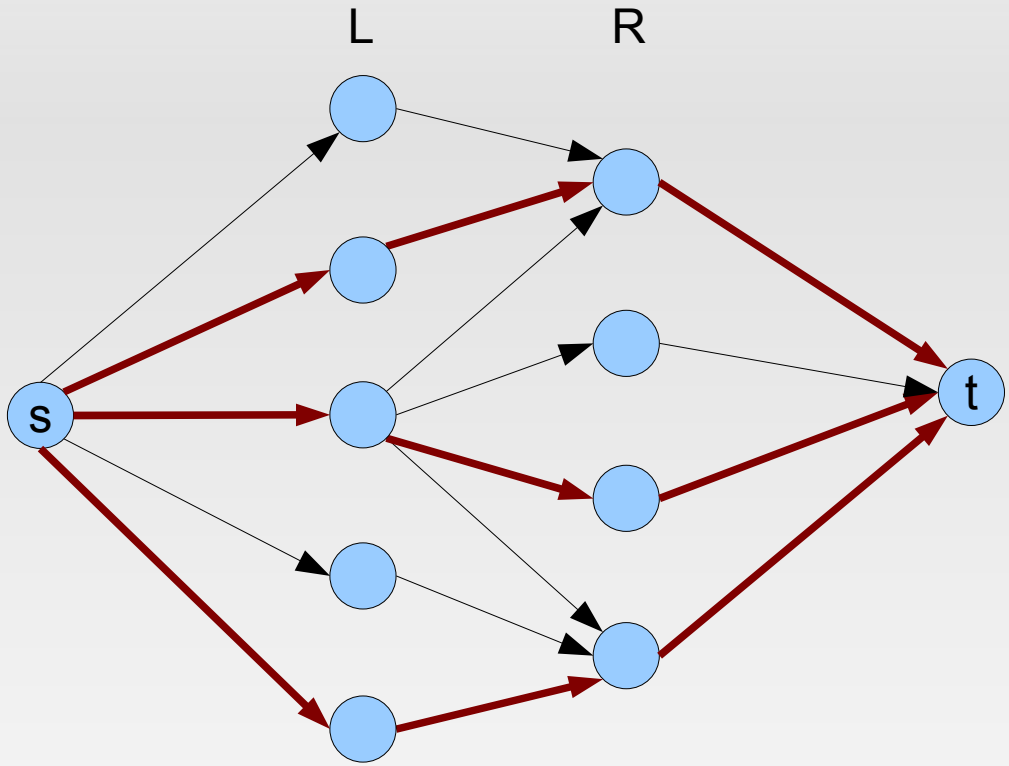


Flow Network

Max Matching as Max Flow



Maximum Bipartite Matching



Maximum Flow

Prove Corollary 26.12

- *Integrality Theorem 26.11*
  - If the capacity function c takes only integral values, then the value |f| of the maximum flow f produced by the Ford-Fulkerson method is an integer (Ex. 26.3-2).
- *Lemma 26.10*
  - Let G be a bipartite graph and let G' its be its corresponding flow network.
  - The following two statements are equivalent:
    - There is a matching M in G such that |M| = k;
    - There is an integer flow f in G' such that |f| = k;

Show that 26.11 + 26.10 ⇒ 26.12 (max match |M| = max flow |f|)