Chap 26 – Maximum Flow

26.1 Flow networks

26.2 The Ford-Fulkerson method

26.3 Maximum bipartite matching

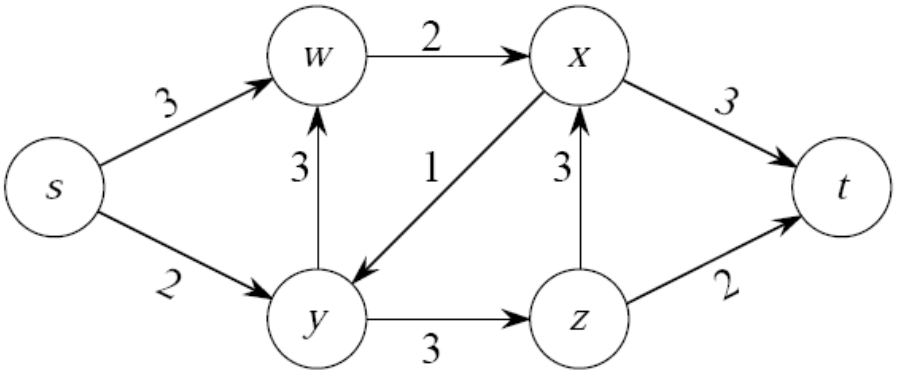
*26.4 Push-relabel algorithms

*26.5 The relabel-to-front algorithm

26.1 Flow networks

- Flow networks
  - $G = (V, E)$ directed
  - Each edge (u, v) has a ***capacity*** $c(u, v) \geq 0$.
  - If $(u, v) \notin E$, then $c(u, v) = 0$.
  - If $(u, v) \in E$, then reverse edge $(v, u) \notin E$
  - ***Source*** vertex s, ***sink*** vertex t

    Assume $s \rightsquigarrow v \rightsquigarrow t$ for all $v \in V$
  - Example



26.1 Flow networks

- Flow networks
  - A **flow** is a function $f : V \times V \to \mathbb{R}$ satisfying
    - Capacity constraint

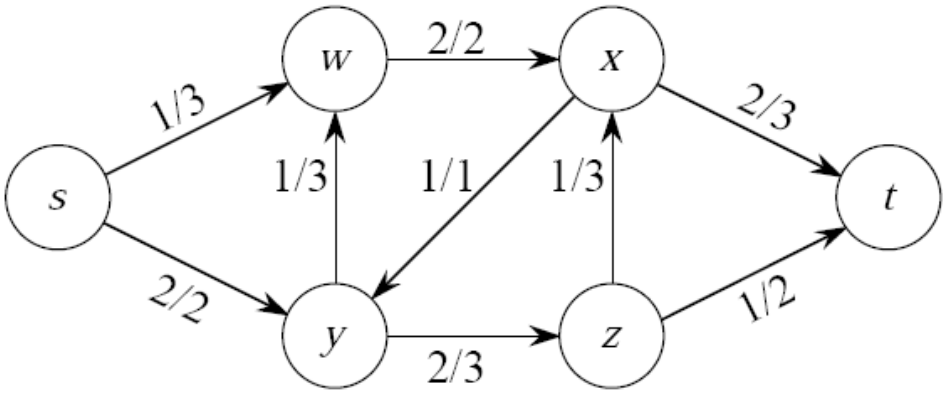
      For all $u, v \in V$, $c(u, v) \geq f(u, v) \geq 0$
    - Flow conservation

      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 flow f

    $= |f| = \underbrace{\sum_{v \in V} f(s, v)}_{\text{flow out of source}} - \underbrace{\sum_{v \in V} f(v, s)}_{\text{flow into source}}$

26.1 Flow networks

- Flow networks
  - Example



Value of flow = 3

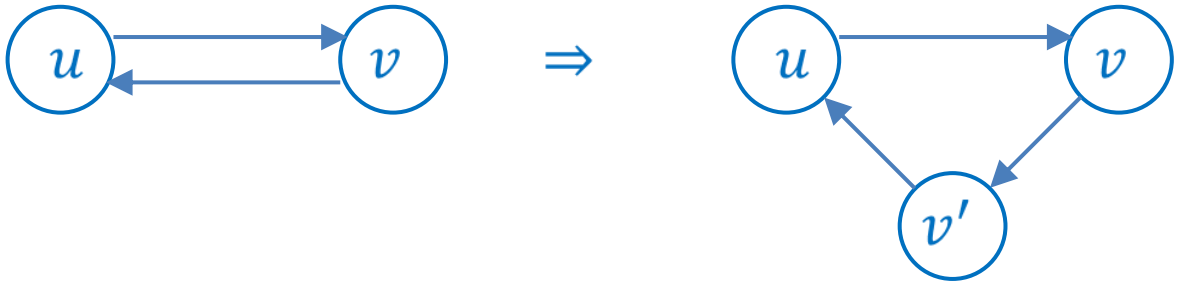
Note that value of flow = 0 is legitimate, so, we want the maximum flow.

  - Maximum-flow problem

Given G, s, t and c, find a flow whose value is maximum.

26.1 Flow networks

- Antiparallel edges
  - The previous definition of a flow network doesn't allow antiparallel edges, i.e. edges (u, v) and (v, u).
  - This restriction may be worked around as follows.
    - Create a new vertex v'
    - Replace (v, u), say, by two edges (v, v') and (v', u)
    - Let $c(v, v') = c(v', u) = c(v, u)$



26.1 Flow networks

- Cuts
  - A cut (S, T) of flow network is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$.
    - Similar to cut used in MST, except that here the graph is directed, and we require $s \in S$ and $t \in T$.
  - The **capacity** of cut (S, T) counts capacities of edges going from S to T and ignores edges in the reverse direction:

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

  - A minimum cut of G is a cut whose capacity is minimum over all cuts of G.

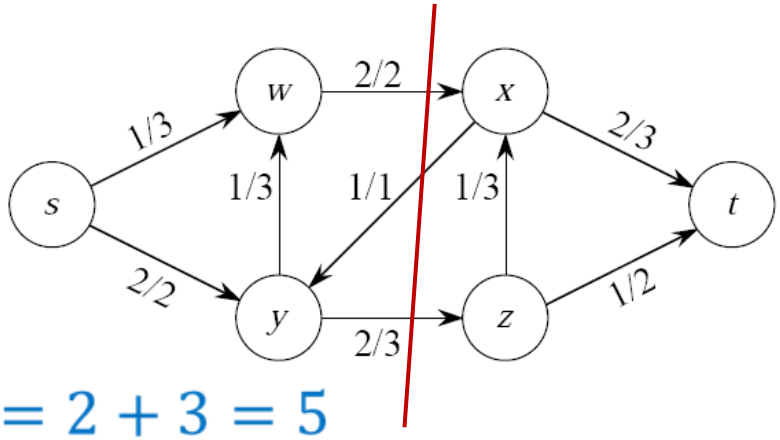
26.1 Flow networks

- Cuts
  - For flow f, the **net flow** across cut (S, T) counts flow on all edges across the cut:

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

  - Example

$S = \{s, w, y\}, T = \{x, z, t\}$



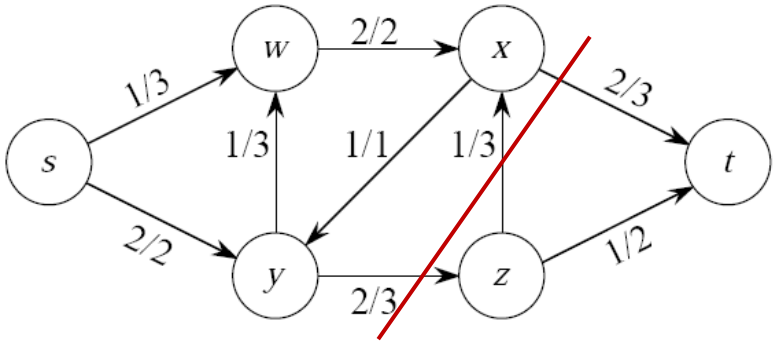
$$c(S,T) = c(w,x) + c(y,z) = 2 + 3 = 5$$

$$f(S,T) = f(w,x) + f(y,z) - f(x,y) = 2 + 2 - 1 = 3$$

26.1 Flow networks

- Cuts
  - Example (Cont'd)

    $S = \{s, w, x, y\}, T = \{z, t\}$



$$c(S,T) = c(x,t) + c(y,z) = 3 + 3 = 6$$

$$f(S,T) = f(x,t) + f(y,z) - f(z,x) = 2 + 2 - 1 = 3$$

Same flow as previous cut, but higher capacity

  - **LEMMA** For any flow f and cut (S,T), $|f| = f(S,T)$

    Intuition

    No matter where you cut the pipes in a network, you'll see the same flow volume coming out of the openings.

26.1 Flow networks

- Cuts
  - **COROLLARY** For any flow f and cut (S, T), $|f| \leq c(S, T)$

    Proof $|f| = f(S, T)$

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

$$\leq \sum_{u \in S} \sum_{v \in T} f(u, v)$$

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

  - Therefore, maximum flow $\leq$ capacity of minimum cut.

    In fact, they are equal. (See later)

26.2 The Ford-Fulkerson method

- Residual network
  - Given a flow f in network $G = (V, E)$

    Define the ***residual capacity*** for $u, v \in V$

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

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

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

    Each edge of G_f admits a positive flow.

26.2 The Ford-Fulkerson method

- Residual network
  - Example

$(x, y) = (u, v)$

$\Rightarrow c_f(x, y) = c(x, y) - f(x, y) = 5 - 2 = 3$

$(y, x) = (u, v) \Rightarrow c_f(y, x) = f(x, y) = 2$

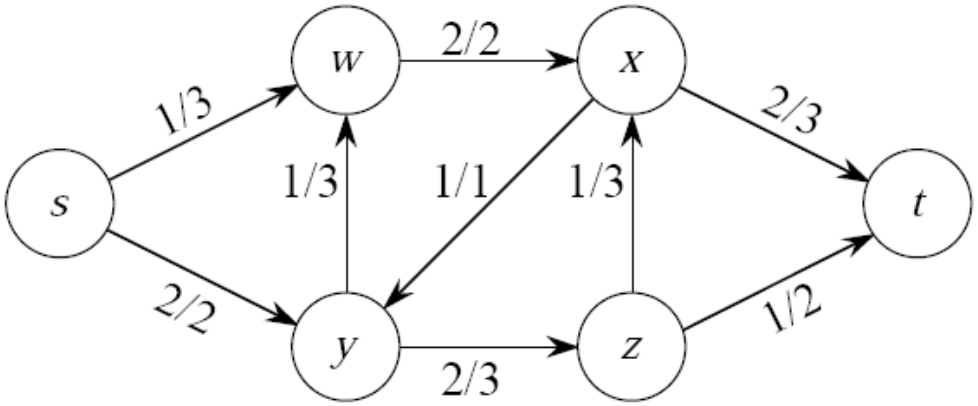
may increase flow by up to 3 units from x to y

may return up to 2 units of flow from y to x

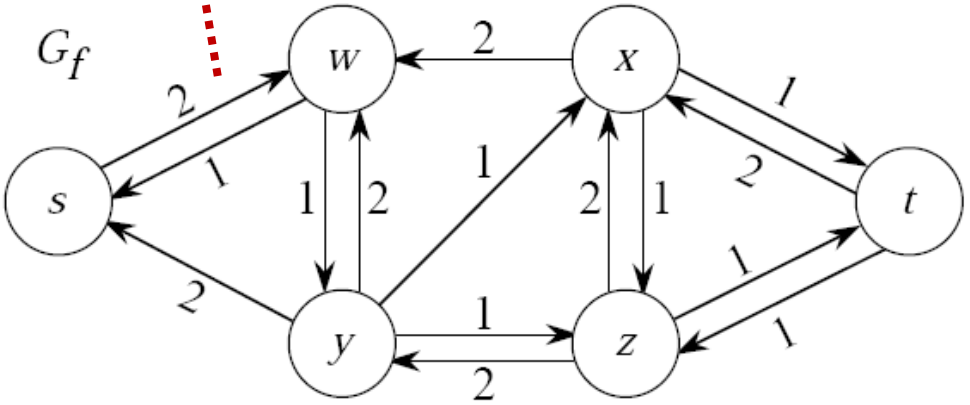
equivalently, may cancel up to 2 units of flow from x to y

26.2 The Ford-Fulkerson method

- Residual network
  - Example



The flow from s to w may increase by at most 2 units and decrease by at most 1 unit.



26.2 The Ford-Fulkerson method

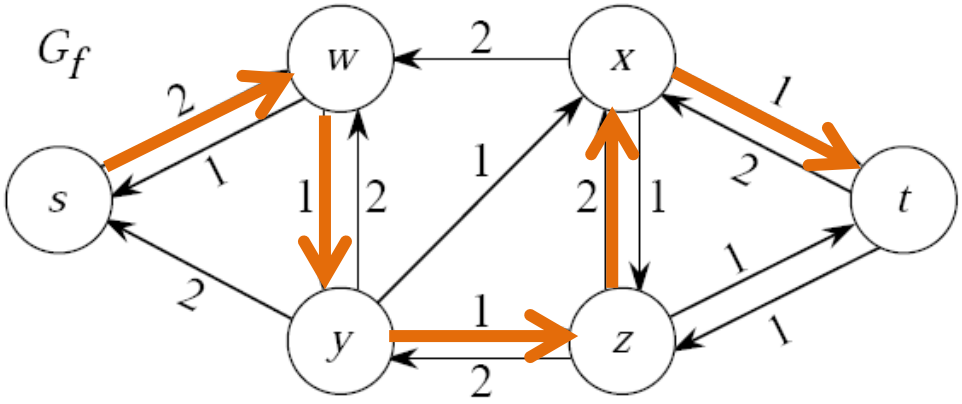
- Residual network
  - The residual network is similar to a flow network, except that it may contain antiparallel edges.
    We can define a flow in a residual network that satisfies the definition of a flow, but w.r.t capacities c_f in G_f.
  - Given flows f in G and f' in G_f, define the **augmentation** of f by f' as a function $V \times V \to \mathbb{R}$:

$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + f'(u,v) - f'(v,u), & \text{if } (u,v) \in E \\ 0, & \text{otherwise} \end{cases}$$

  - **LEMMA**
    $|f \uparrow f'|$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$

26.2 The Ford-Fulkerson method

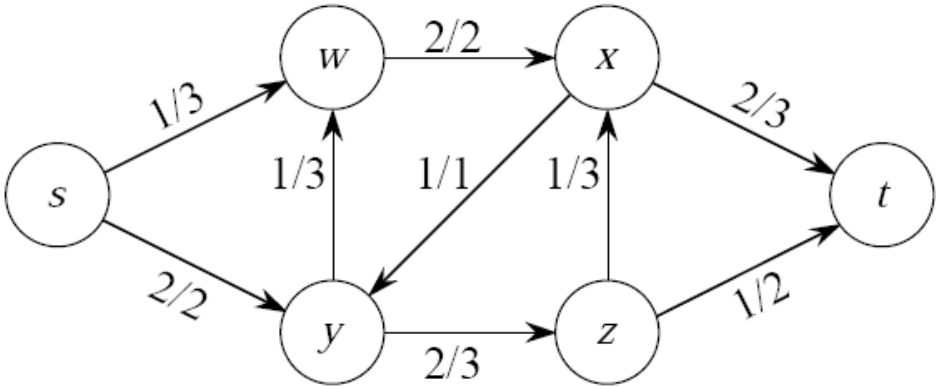
- Augmenting path
  - An augmenting path is a simple path $s \rightsquigarrow t$ in G_f.
  - Example



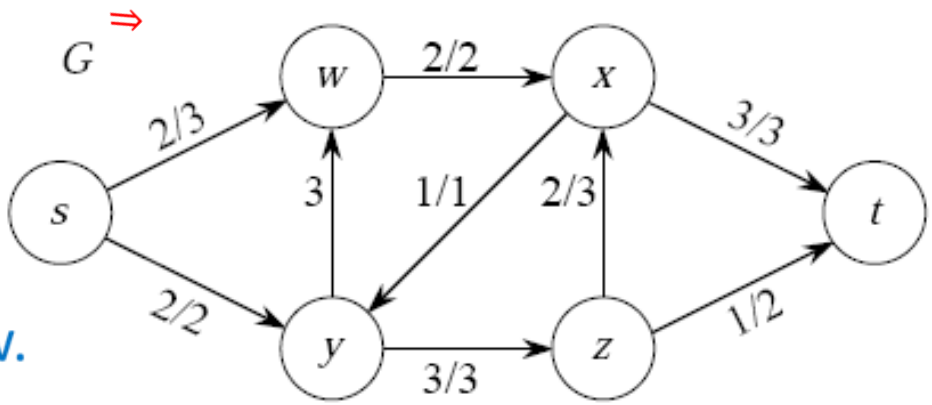
We can increase the flow through each edge of the path by 1 unit without violating a capacity constraint, since the smallest residual capacity on this path is 1.

26.2 The Ford-Fulkerson method

- Augmenting path
  - Example (Cont'd)



value of flow = 3



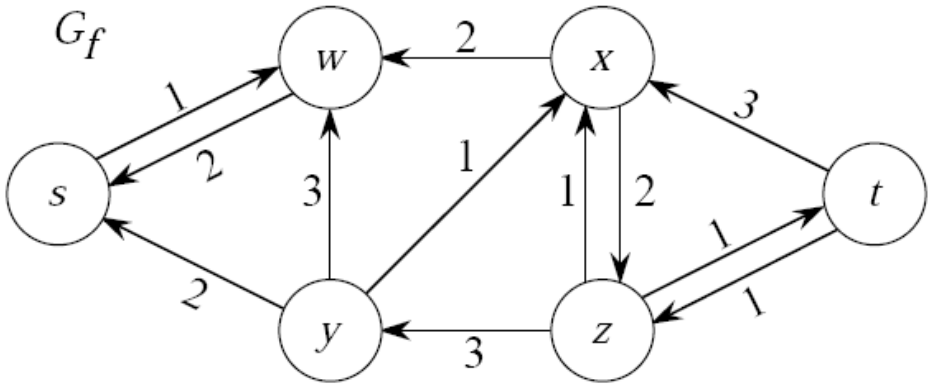
value of flow = 4

This is a maximum flow.

26.2 The Ford-Fulkerson method

- Augmenting path
  - Example (Cont'd)

    Observe that, for this augmented flow network G, the corresponding residual network G_f is:



Note that there is no augmenting path in this G_f, which implies that the flow shown in G is a maximum flow.

26.2 The Ford-Fulkerson method

- Augmenting path
  - The residual capacity of an augmenting path p is

    $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$
  - **LEMMA**

    Let p be an augmenting path in G_f.

    Define $f_p : V \times V \to \mathbb{R}$

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

    Then, f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$
  - **COROLLARY**

    $f \uparrow f_p$ is a flow in G with value $|f \uparrow f_p| = |f| + |f_p| > |f|$

26.2 The Ford-Fulkerson method

- Max-flow min-cut theorem
  - **THEOREM**

    The following statements are equivalent.

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

    Proof

    $(1) \Rightarrow (2)$

    Suppose not. Let p be an augmenting path in G_f.

    Then, $|f \uparrow f_p| = |f| + |f_p| > |f|$.

    A contradiction.

26.2 The Ford-Fulkerson method

- Max-flow min-cut theorem
  - **THEOREM** (Cont'd)

    $(2) \Rightarrow (3)$ Suppose G_f contains no augmenting paths.

    Define

    $S = \{u \in V : \text{there exists a path } s \rightsquigarrow u \text{ in } G_f\}$

    $T = V - S$

    Then, $t \in T$; otherwise, there is an augmenting path.

    Therefore, (S, T) is a cut.

    Consider $u \in S$ and $v \in T$

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

      $\because f(u, v) < c(u, v) \Rightarrow c_f(u, v) > 0 \Rightarrow (u, v) \in E_f$

      $\Rightarrow s \rightsquigarrow u \rightsquigarrow v \Rightarrow v \in S$

26.2 The Ford-Fulkerson method

- Max-flow min-cut theorem
  - **THEOREM** (Cont'd)
    - If $(v,u) \in E$, then $f(v,u) = 0$

      $\because c_f(u,v) = f(v,u) > 0 \Rightarrow (u,v) \in E_f \Rightarrow v \in S$
    - If $(u,v), (v,u) \notin E$, then $f(v,u) = f(u,v) = 0$

    Then,

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

$$= \sum_{u \in S}\sum_{v \in T} c(u,v) - \sum_{u \in S}\sum_{v \in T} 0 = c(S,T)$$

    By lemma, $|f| = f(S,T) = c(S,T)$

26.2 The Ford-Fulkerson method

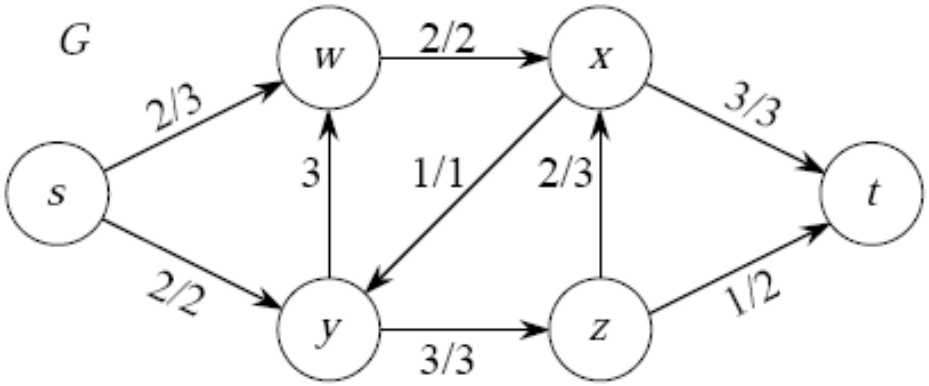
- Max-flow min-cut theorem
  - **THEOREM** (Cont'd)

    $(3) \Rightarrow (1)$

    By corollary, $|f| \leq c(S,T)$ for all cuts (S,T)

    Therefore, $|f| = c(S,T) \Rightarrow f$ is a maximum flow in G
  - Example (Cont'd)

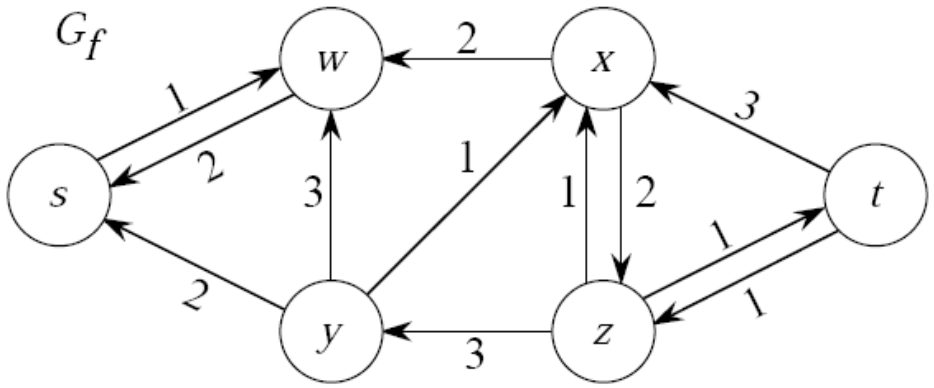
    Recall that $|f| = 4$, where f is a maximum flow in G:



26.2 The Ford-Fulkerson method

- Max-flow min-cut theorem
  - Example (Cont'd)

    Also, the corresponding G_f has no augmenting paths:



Now, let

$S = \{u \in V : \text{there exists a path } s \rightsquigarrow u \text{ in } G_f\} = \{s, w\}$

$T = V - S = \{x, y, z, t\}$

Then, (S, T) is a minimum cut with

$c(S, T) = c(s, y) + c(w, x) = 2 + 2 = 4$

26.2 The Ford-Fulkerson method

- Ford-Fulkerson algorithm
  - Keep augmenting flow along an augmenting path until there is no augmenting path.
  - FORD-FULKERSON(G, s, t)

    for all $(u, v) \in G.E$

    $(u, v).f = 0$

    while there is an augmenting path p in G_f

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

    for each (u, v) in p

    if $(u, v) \in E$ **then** $(u, v).f = (u, v).f + c_f(p)$

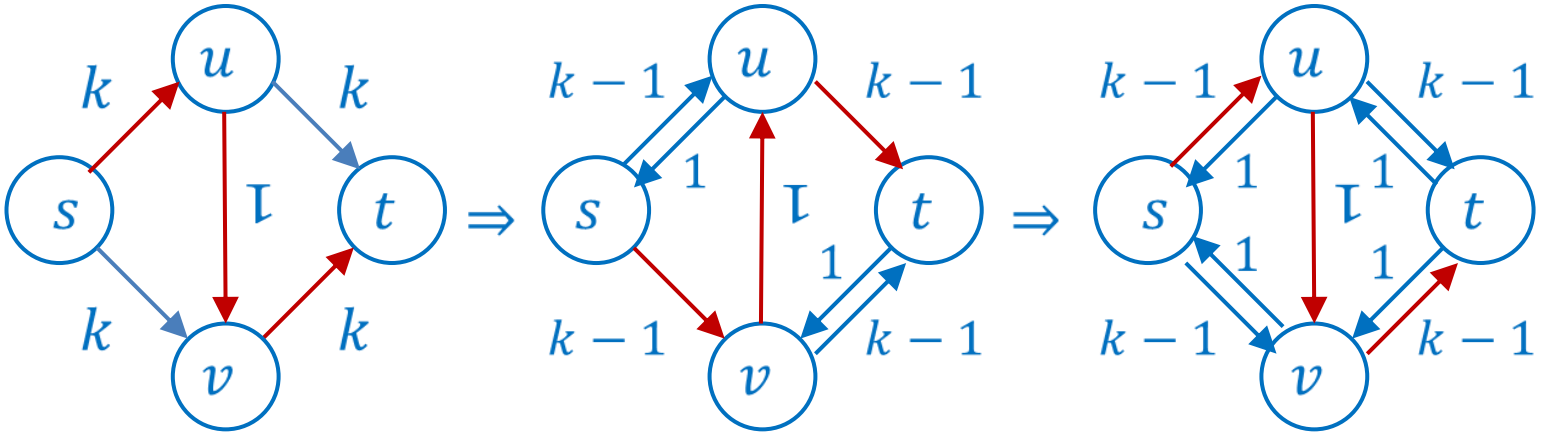
    else $(v, u).f = (v, u).f - c_f(p)$

26.2 The Ford-Fulkerson method

- Ford-Fulkerson algorithm
  - Time: $O(E|f^*|)$, where f^* is a maximum flow, assuming that the capacities are integers.
    - The **while** loop is executed at most $|f^*|$ times. ($\because$ the flow value increases by at least one unit in each iteration)
    - Keep a data structure for the graph $G' = (V, E')$, where $E' = \{(u, v) : (u, v) \in E \text{ or } (u, v) \in E\}$
      Then, $E_f = \{(u, v) \in E' : c_f(u, v) > 0\}$
      Thus, an augmenting path in G_f, if any, can be found in $O(V + E') = O(E)$ time, using depth- or breath-first search on G'.

26.2 The Ford-Fulkerson method

- Ford-Fulkerson algorithm
  - Note that this running time is not polynomial in input size. It depends on $|f^*|$, which is not a function of $|V|$ and $|E|$.
  - If $|f^*|$ is large, the performance may be poor.
  - Example – $|f^*| = 2k$ augmentations



$G, |f^*| = 2k$ G_f

26.2 The Ford-Fulkerson method

- Edmonds-Karp algorithm
  - Ford-Fulkerson algorithm doesn't specify how to find an augmenting path. In particular, it doesn't specify which one to use if more than one is available.
  - **Edmonds-Karp algorithm**

    A variant of Ford-Fulkerson algorithm that requires the augmenting path be a shortest path $s \rightsquigarrow t$ in G_f.

    This can be found by a BFS, since all edge weights $= 1$
  - Edmonds-Karp algorithm takes a time in $O(VE^2)$.

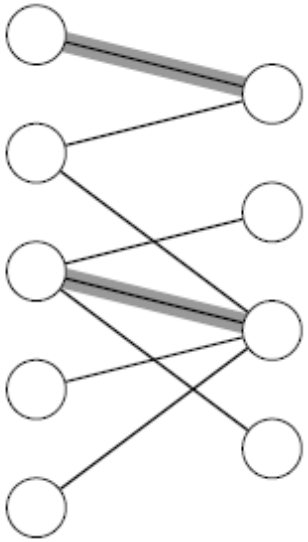
    See book for a proof.

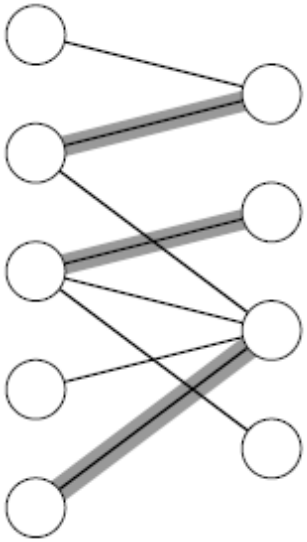
26.3 Maximum bipartite matching

- Maximum bipartite matching
  - An undirected graph $G = (V, E)$ is ***bipartite*** if we can partition $V = L \cup R$ such that all edges in E go between L and R.
  - A ***matching*** is a subset of edges $M \subseteq E$ such that for all $v \in V$, at most one edge of M is incident on v.
  - A vertex is *matched* if some edge of M is incident on it; otherwise, it is *unmatched*.
  - A maximum matching M is a matching of maximum cardinality, i.e. $|M| \geq |M'|$ for any matching M'.

26.3 Maximum bipartite matching

- Maximum bipartite matching
  - Example: matching maximum matching



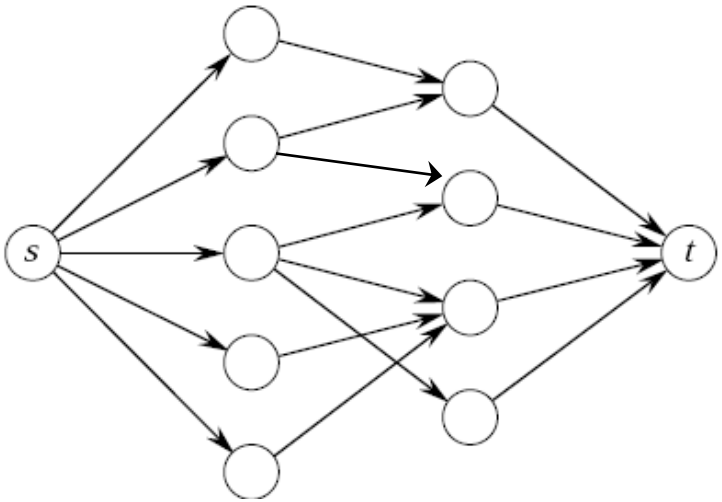


  - Problem

    Given a bipartite graph, find a maximum matching.

26.3 Maximum bipartite matching

- Maximum bipartite matching
  - Given a bipartite graph $G = (V, E)$, where $V = L \cup R$, define flow network $G' = (V', E')$
    - $V' = V \cup \{s, t\}$
    - $E' = \{(s, u) : u \in L\} \cup E \cup \{(v, t) : v \in R\}$
    - $c(u, v) = 1$ for all $(u, v) \in E'$



26.3 Maximum bipartite matching

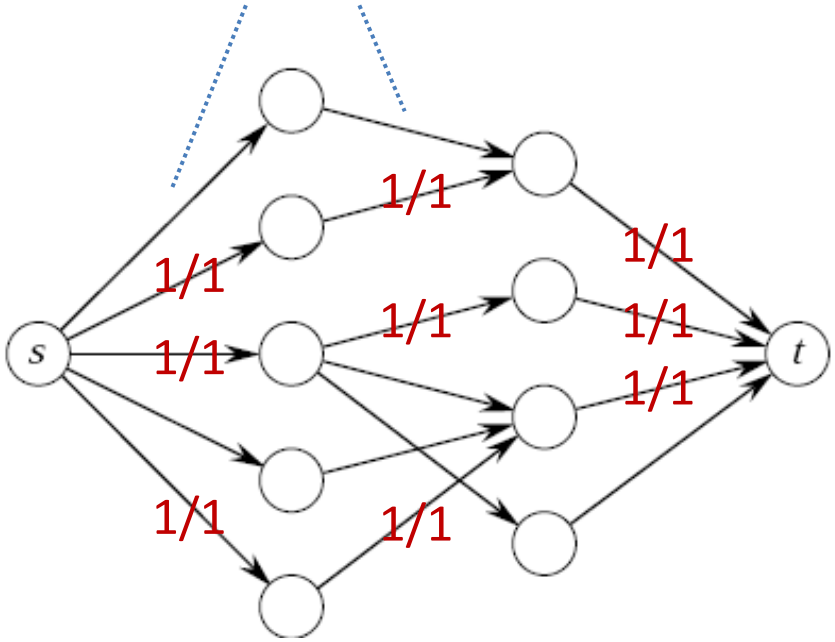
- Maximum bipartite matching
  - **THEOREM**

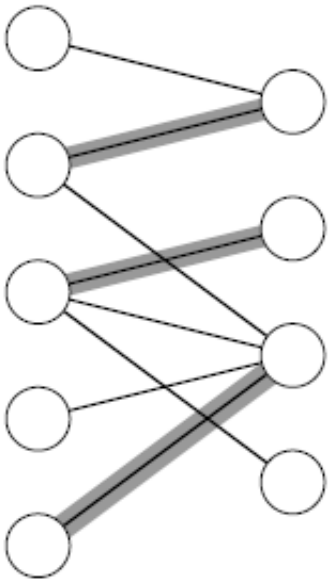
    Let M be a maximum matching in a bipartite graph G. Let f be a maximum flow in the corresponding flow network G'. Then, $|M| = |f|$
  - MBM(G)
    1. Create the flow network G'
    2. Run Ford-Fulkerson algorithm on G'
    3. Obtain a maximum matching M directly from the maximum flow f found

26.3 Maximum bipartite matching

- Maximum bipartite matching
  - Example

all other edges, 0/1





maximal flow f in G' ⇒ maximum matching M in G

26.3 Maximum bipartite matching

- Maximum bipartite matching
  - Time: $O(VE)$
    - Since each vertex in V has at least one incident edge, we have $|E| \geq |V|/2$.
      $\therefore |E| \leq |E'| = |E| + |V| \leq 3|E| \Rightarrow |E'| = \Theta(|E|)$
    - For any matching M, $|M| \leq \min(L, R) = O(V)$
    - Thus, $|f^*| = O(V)$ and Ford-Fulkerson algorithm runs in $O(E'|f^*|) = O(VE)$ time.
      N.B. Edmonds-Karp algorithm takes a time in $O(V'E'^2) = O((V+2)E^2) = O(VE^2)$.