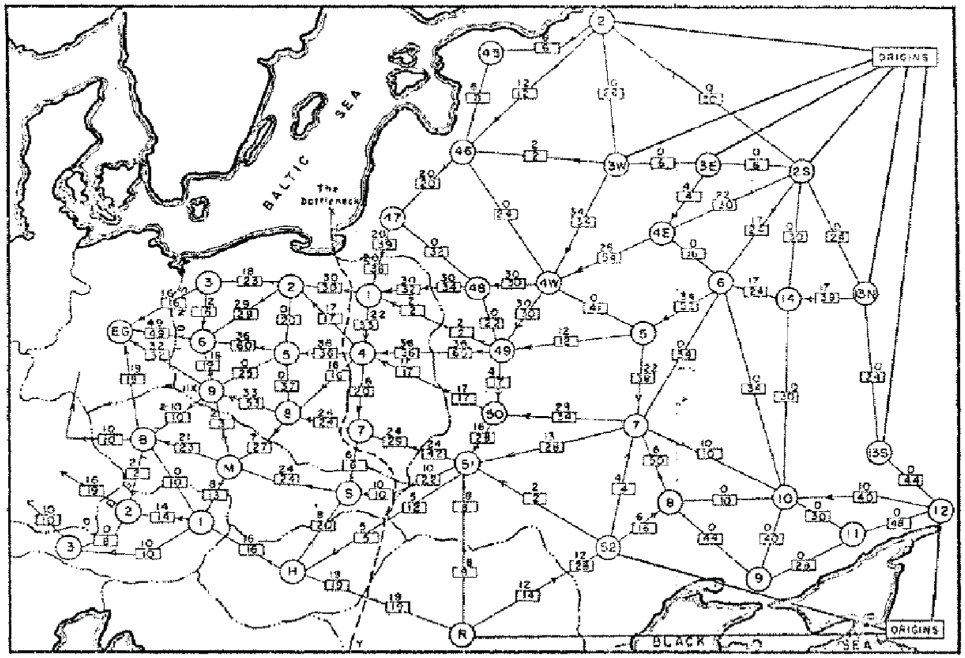
Maximum Flow

Chapter 26 from textbook

Maximum flow

Soviet Rail Network, 1955



Reference: *On the history of the transportation and maximum flow problems.*
Alexander Schrijver in Math Programming, 91: 3, 2002.

•Material coursing through a system from a source to a sink

Flow graphs

- A common scenario is to use a graph to represent a “flow network” and use it to answer questions about material flows
- Flow is the rate that material moves through the network
- Each directed edge is a conduit for the material with some stated capacity
- Vertices are connection points but do not collect material
  - Flow into a vertex must equal the flow leaving the vertex, flow conservation

Sample networks

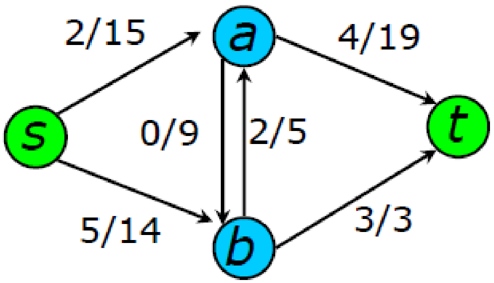
Network	Nodes	Arcs	Flow
communication	telephone exchanges, computers, satellites	cables, fiber optics, microwave relays	voice, video, packets
circuits	gates, registers, processors	wires	current
mechanical	joints	rods, beams, springs	heat, energy
hydraulic	reservoirs, pumping stations, lakes	pipelines	fluid, oil
financial	stocks, companies	transactions	money
transportation	airports, rail yards, street intersections	highways, railbeds, airway routes	freight, vehicles, passengers
chemical	sites	bonds	energy

Flow concepts

- Source vertex s
  - where material is produced
- Sink vertex t
  - where material is consumed
- For all other vertices – what goes in must go out
  - Flow conservation
- **Goal: determine maximum rate of material flow from source to sink**

Max flow problem

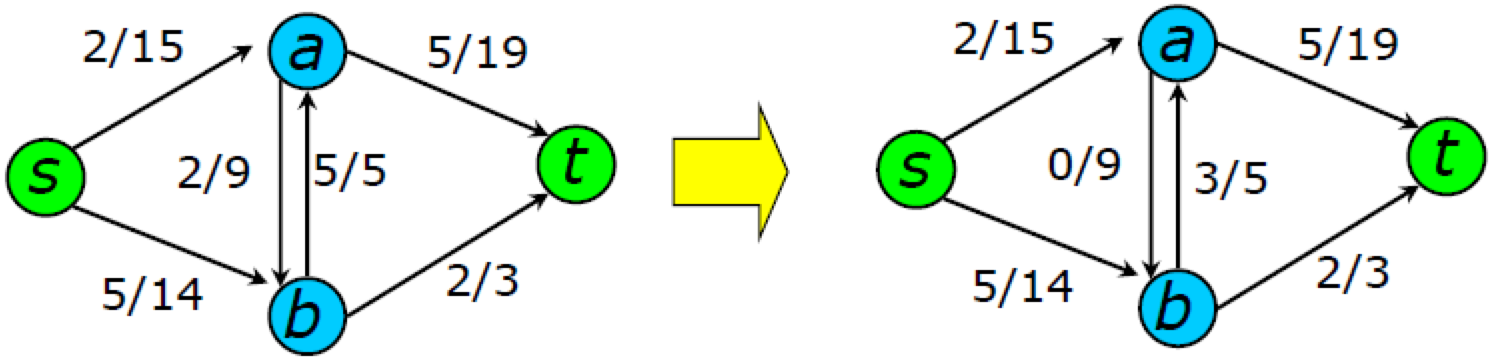
- Graph G=(V,E) – a **flow network**
  - Directed, each edge has **capacity** $c(u,v) \geq 0$
  - Two special vertices: **source** s, and **sink** t
  - For any other vertex v, there is a path $s \to \ldots \to v \to \ldots \to t$
- **Flow** – a function $f : V \times V \to \mathbf{R}$
  - Capacity constraint: For all $u, v \in V$: $f(u,v) \leq c(u,v)$
  - Skew symmetry: For all $u, v \in V$: $f(u,v) = -f(v,u)$
    - If (u,v) is not an edge f(u,v)=0 and c(u,v)=0
  - Flow conservation: For all $u \in V - \{s, t\}$: total incoming and outgoing flow of any node u are equal



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

Cancellation of flows

- We would like to avoid two positive flows in opposite directions between the same pair of vertices
  - Such flows cancel (maybe partially) each other due to skew symmetry

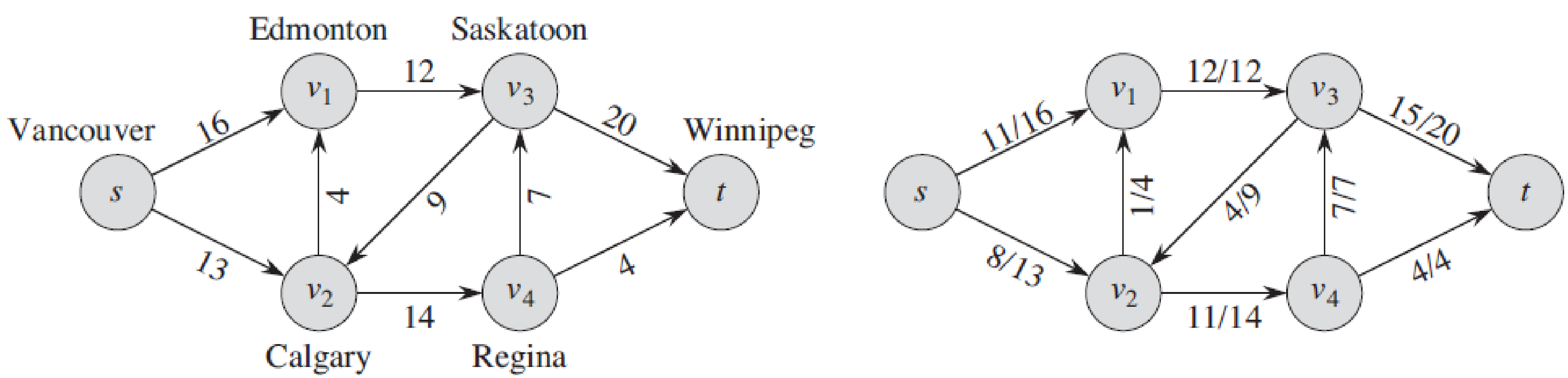


Definition of flows

The non-negative quantity f(u, v) is called the **flow** from vertex u to vertex v. The value |f| of a flow f is defined as

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

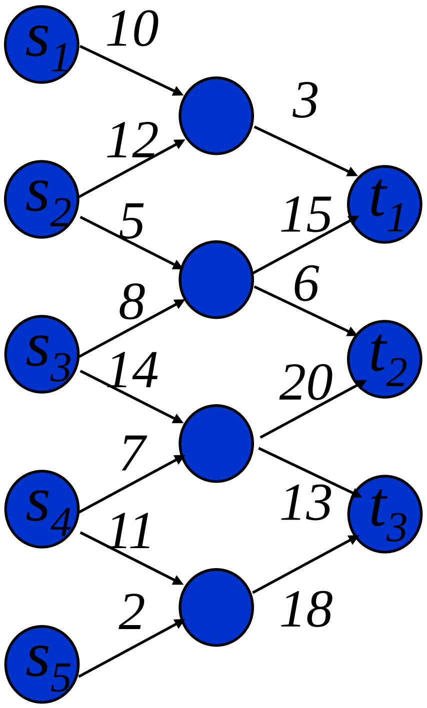
- **Example**

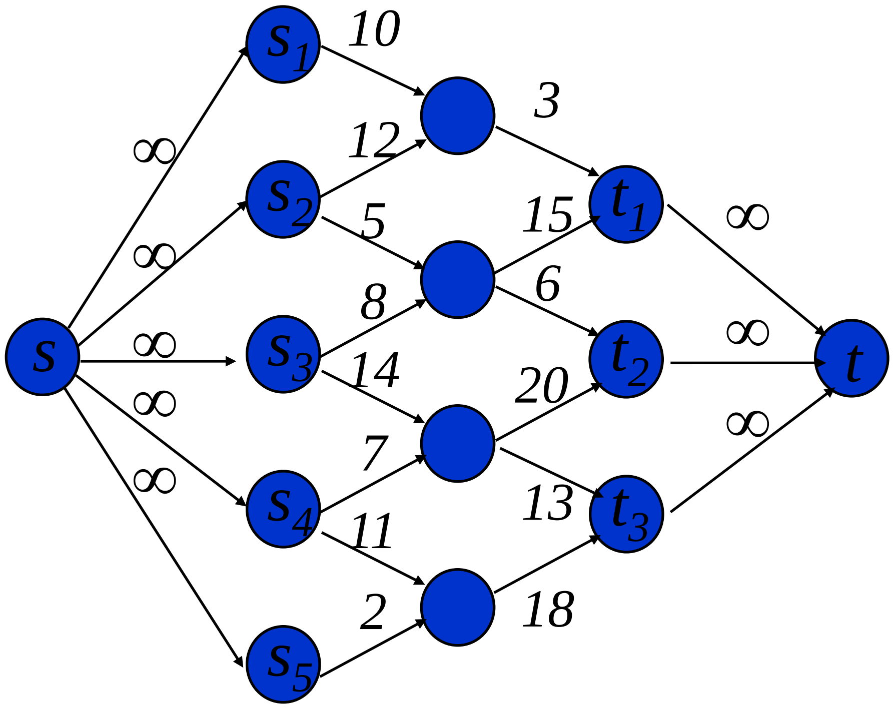


- Value of the flow is 19
- Total incoming flow at any node is equal to the total outgoing flow at any node

Handling multiple sources/sinks

Introduce supersource s and supersink t





Ford-Fulkerson method

- **The maximum-flow problem:** given a flow network G with source s and sink t, we wish to find a flow f of maximum value.
- Important concepts:
  - residual networks
  - augmenting paths
  - cuts

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

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

Residual capacity

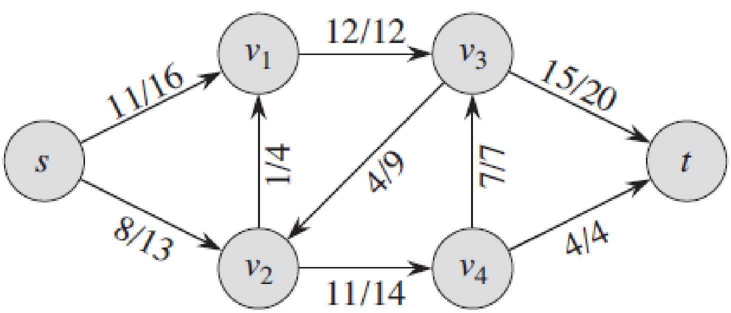
- Given a flow network G and a flow f, the residual network G_f consists of edges with capacities that represents how we can change the flow (admit more flow) on this edge.
- Let f be a flow in G = (V, E) with source s and sink t. For any pair of vertices u, v $\in$ V, the amount of additional flow we can push from u to v before exceeding the capacity c(u, v) is the **residual capacity** of (u, v), given by

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

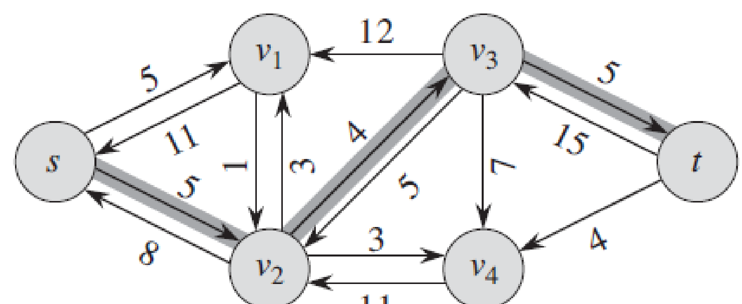
- Example
  - If c(u, v) = 16 and f(u, v) = 11, then c_f(u, v) = 16 – 11 = 5.
  - If c(u, v) = 0 and f(v, u) = 4, then c_f(u, v) = 0 – (-4) = 4

Residual Network

- Given a flow network $G = (V, E)$ and a flow f, the **residual network** of G induced by f is $G_f = (V, E_f)$, where
  - $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$.
- **Example**



Original network



Residual network

Maximum number of edges in a residual network is $|E_f| \leq 2|E|$

Flow augmentation

- A flow in a residual network provides a roadmap for adding flow to the original network
- If f is a flow in G and f' is a flow in corresponding residual network G_f, we denote $(f \uparrow f')$, the augmentation of flow f by f' to be a function

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

  - The intuition is we increase the flow on (u,v) by f'(u,v) but decrease it by f'(u,v) because pushing flow on the reverse edge in the residual network signifies decreasing the flow in the original network

Flow augmentation

- ***Lemma 26.1*** Let G = (V, E) be a network with source s and sink t, and let f be a flow in G. Let G_f be the residual network of G induced by f, and let f' be a flow in G_f. Then, the flow sum f ↑f' (defined by (f ↑f')(u, v) = f (u, v) + f' (u, v)-f'(v,u)) is a flow in G with value |f ↑ f'| = |f | + |f'|.

  Proof. We must verify that skew symmetry, the capacity constraints, and flow conservation are obeyed.

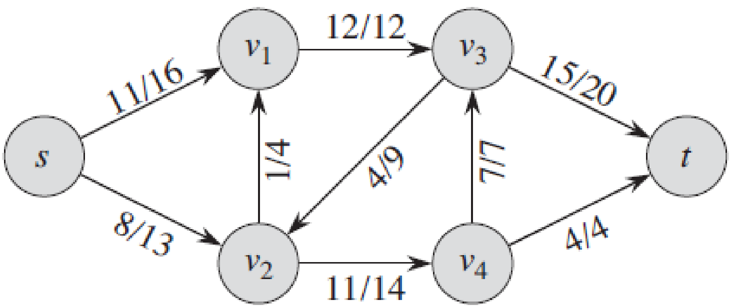
  skew symmetry:

  $(f \uparrow f')(u, v) = f(u, v) + f'(u, v) - f'(v,u) = -f(v, u) - f'(v, u) + f'(u,v)$

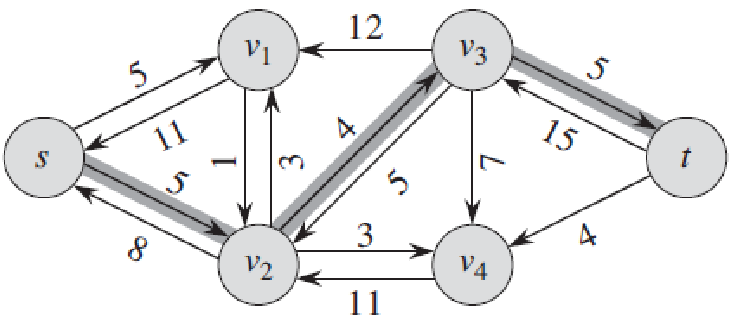
  $= -(f(v, u) + f'(v, u) - f'(u,v)) = -(f \uparrow f')(v, u).$

Augmenting path

- Given a flow network $G = (V, E)$ and a flow f, an augmenting path p is a simple path from s to t in the residual network G_f.
- **Example**



Original network



Residual network

The shaded path in the residual network is an augmenting path

Augmenting path

- In the above residual network, path $s \to v_2 \to v_3 \to t$ is an augmenting path.
- We can increase the flow through each edge of this path by up to 4 units without violating a capacity constraint since the smallest residual capacity on this path is $c_f(v_2, v_3) = 4$.
- Residual capacity of an augmenting path

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

Lemma 26.2 *Let $G = (V, E)$ be a network, let f be a flow in G, and let p be an augmenting path in G_f. Define a function $f_p: V \times V \to \mathbf{R}$ by*

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

Then, f_p is a flow in G_f with value $|f_p| = c_f(p)$.

Residual network and augmenting path

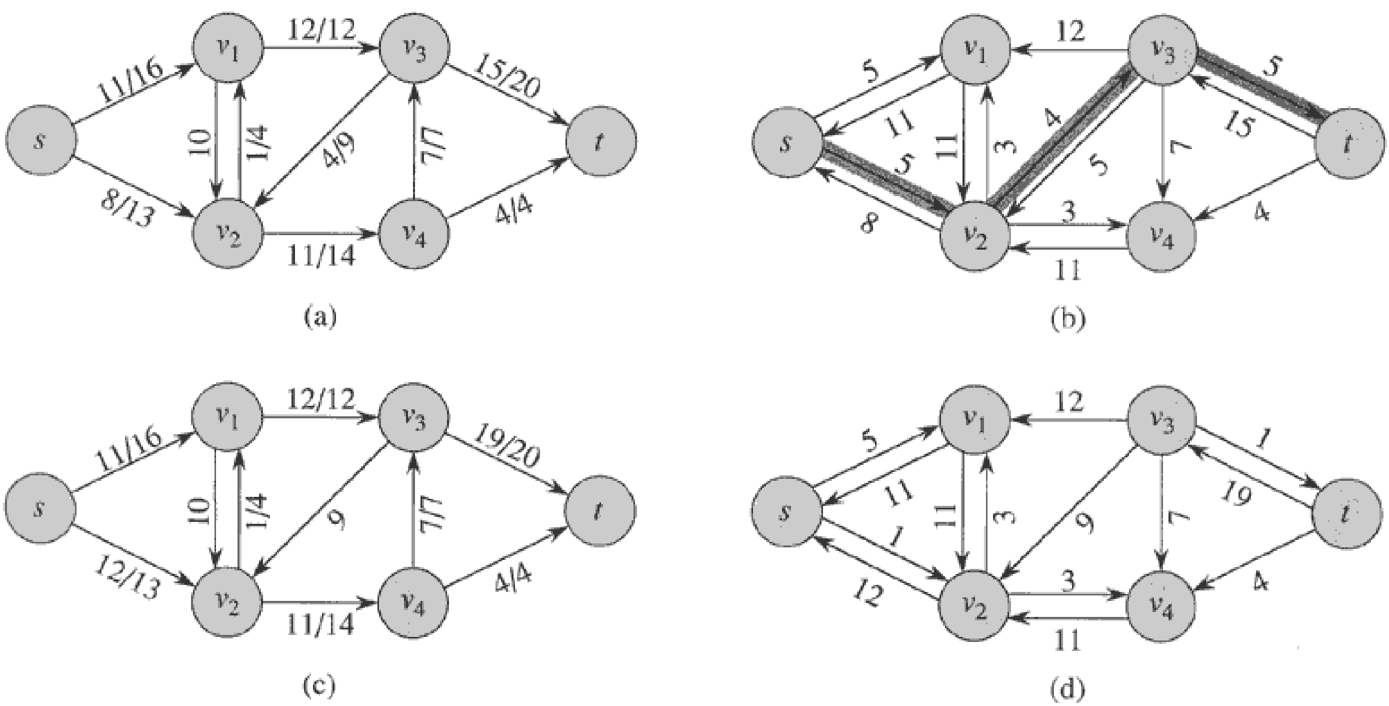


Figure 26.3 (**a**) The flow network G and flow f of Figure 26.1(b). (**b**) The residual network G_f with augmenting path p shaded; its residual capacity is $c_f(p) = c(v_2, v_3) = 4$. (**c**) The flow in G that results from augmenting along path p by its residual capacity 4. (**d**) The residual network induced by the flow in (c).

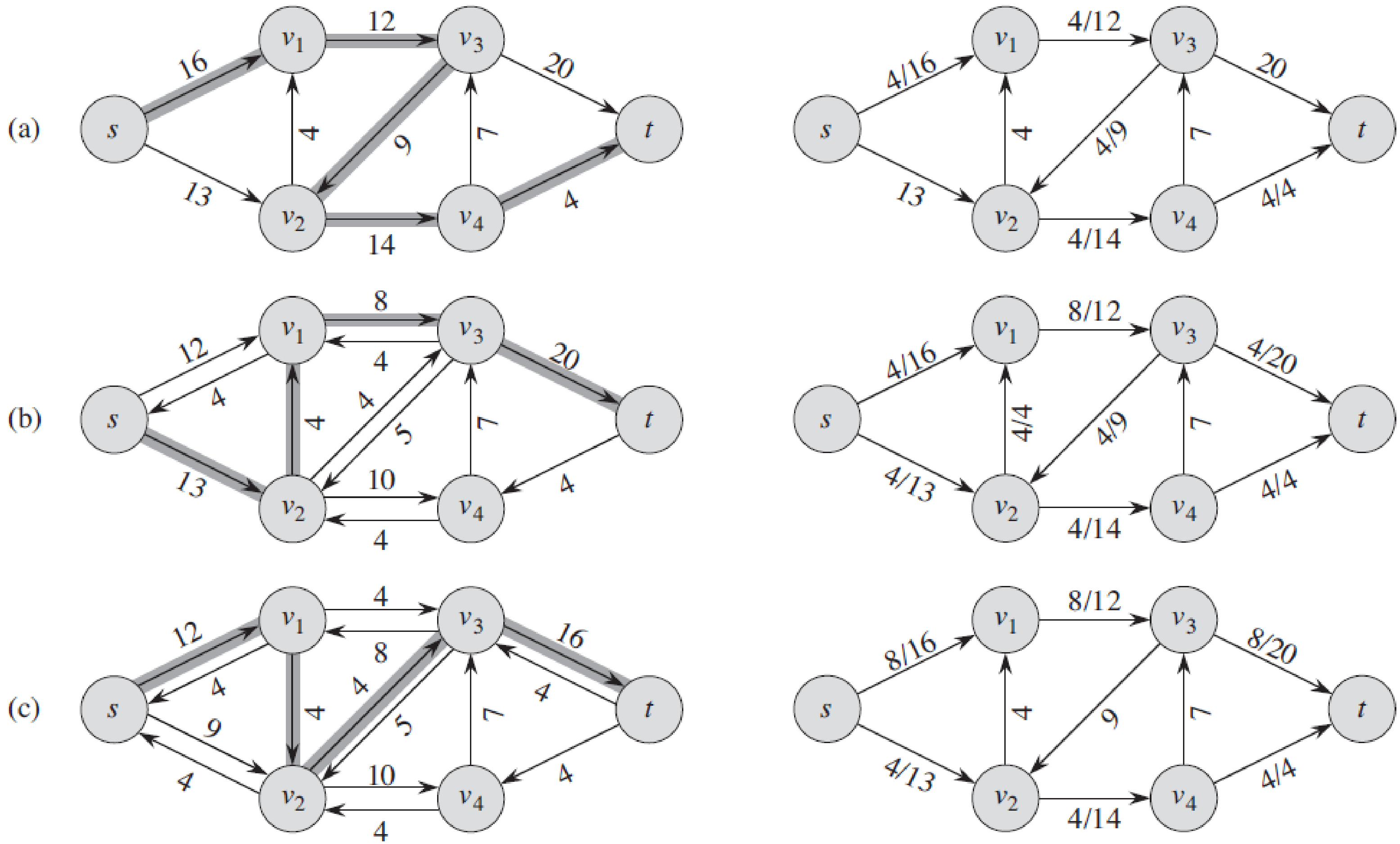
The Ford-Fulkerson method

```
Ford-Fulkerson(G=(V,E),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 residual
  network G_f do
4    c_f = min{c_f(u,v): (u,v) is in p}
5    for each edge (u,v) in p do
6        if (u,v) ∈ E
7            f(u,v) ← f(u,v) + c_f
7        else
8            f(v,u) ← f(u,v) - c_f

8 return f
```

The algorithms based on this method differ in how they choose p in step 3.
If chosen poorly the algorithm might not terminate.

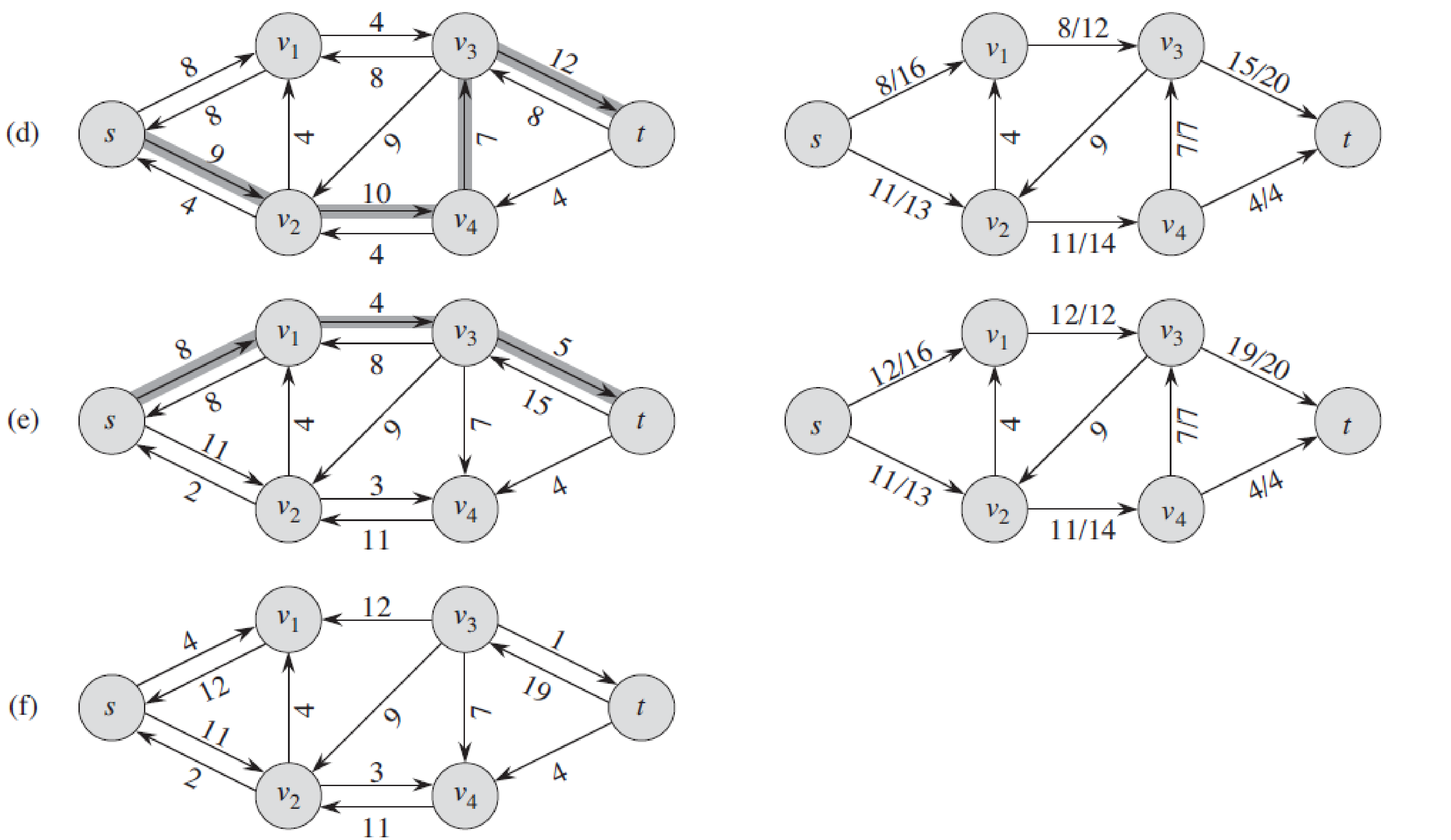
Execution of Ford-Fulkerson method (1)



Left Side = Residual Graph

Right Side = Augmented Flow

Execution of Ford-Fulkerson method (2)



Left Side = Residual Graph *Right Side = Augmented Flow*

Cuts

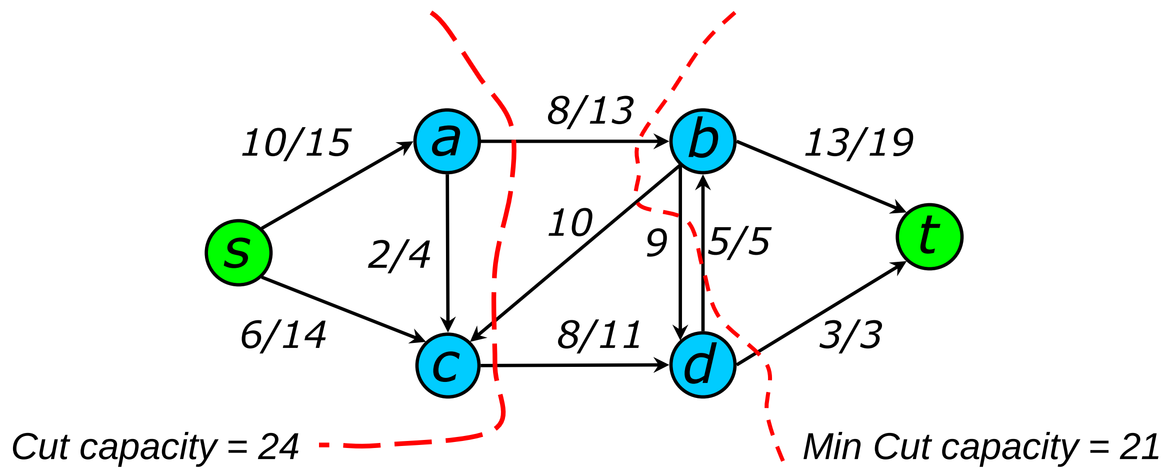
- A **cut** is a partition of V into S and T = V – S, such that $s \in$ S and $t \in$ T
- The **net flow** (f(S,T)) through the cut is the sum of flows f(u,v), where $s \in$ S and $t \in$ T
  - Includes negative flows back from T to S

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

- For a given flow f, the net flow across any cut is the same and is equal to |f|
- The **capacity** (c(S,T)) of the cut is the sum of capacities c(u,v), where $s \in$ S and $t \in$ T

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

- **Minimum cut** – a cut with the smallest capacity of all cuts.



Cuts

- For a given flow f, the net flow across any cut is the same and is equal to |f|
- Therefore, vale of the maximum flow in a network is bounded from above by the capacity of a minimum cut of the network
- The important **max-flow min-cut theorem** says that the value of the maximum flow in fact is equal to the capacity of a minimum cut

Max Flow / Min Cut Theorem

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

1. f is a maximum flow in G

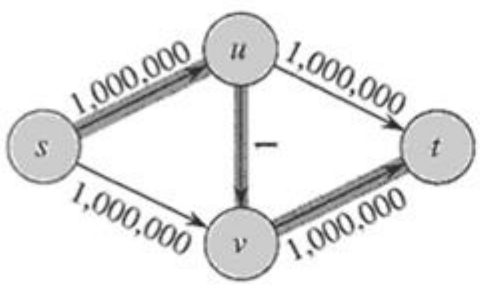
2. The residual network G_f contains no augmenting path

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

- That means Maximum flow in a network is same as the capacity of the minimum cut of the network

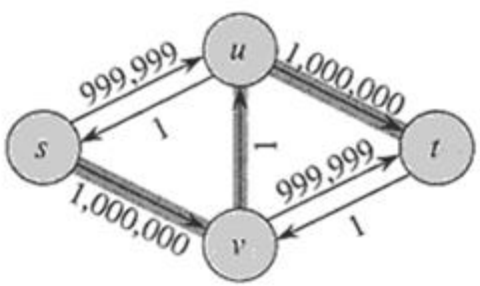
Worst Case Running Time

- Assuming integer flow (that means each f(u,v) an integer)
- Each augmentation increases the value of the flow by some positive amount.
- Augmentation can be done in O(E).
  - This includes BFS/DFS for finding a path and updating flows
- Total worst-case running time O(E|f*|), where f* is the max-flow found by the algorithm (increase flow by 1 in each iteration)
- Example of worst case:



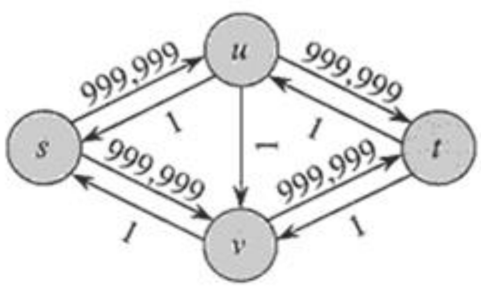
(a)

Augmenting path of 1



(b)

Resulting Residual Network



(c)

Resulting Residual Network

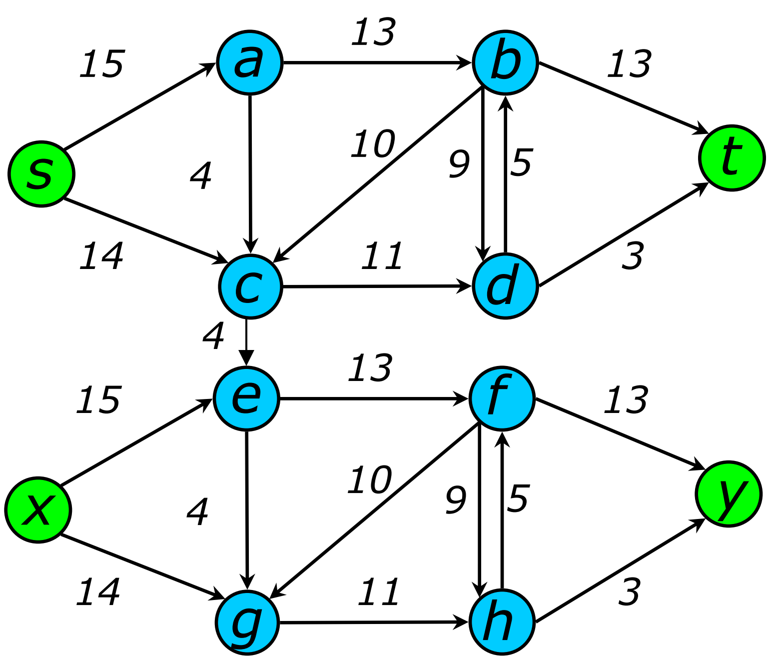
Edmonds Karp

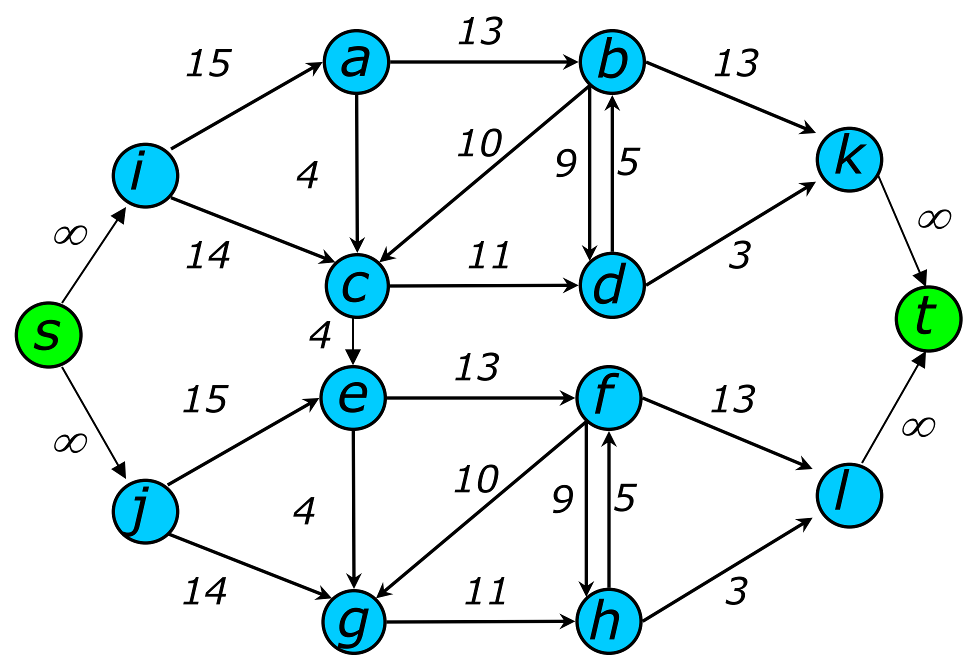
Take **shortest path** (in terms of number of edges) as an augmenting path – Edmonds-Karp algorithm

- How do we find such a shortest path?
  - Assign unit weight to each edge of the Residual network and simply perform BFS
- The number of augmentations (iterations of the algorithm) is in this case can be shown to be at most $O(VE)$
- Each iteration cost O(E)
- Thus, running time $O(VE^2)$
  - Skipping the proof here
- There is even a better method called push-relabel, $O(V^2E)$ runtime (we will not study this)

Multiple Sources or Sinks

- What if you have a problem with more than one source and more than one sink?
- Modify the graph to create a single supersource and supersink



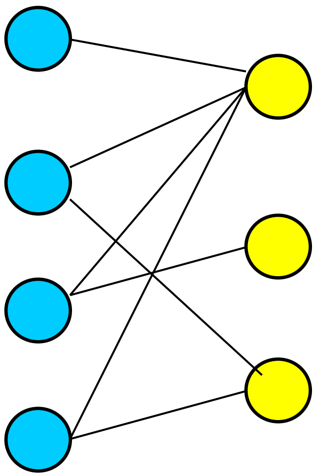


Application – Bipartite Matching

- Example – given a community with *n* men and *m* women
- Assume we have a way to determine which couples (man/woman) are compatible for marriage
  - E.g. (Joe, Susan) or (Fred, Susan) but not (Frank, Susan)
- Problem: Maximize the number of marriages
  - No polygamy allowed
  - Can solve this problem by creating a flow network out of a bipartite graph

Bipartite Graph

- A bipartite graph is an undirected graph G=(V,E) in which V can be partitioned into two sets V_1 and V_2 such that $(u,v) \in E$ implies either $u \in V_1$ and $v \in V_2$ or vice versa.
- That is, all edges go between the two sets V_1 and V_2 and not within V_1 and V_2.

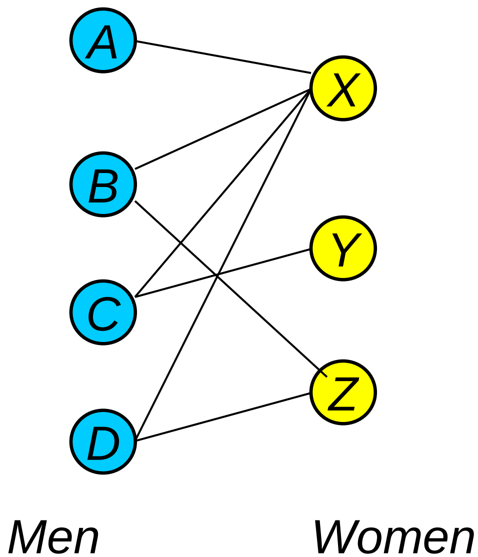


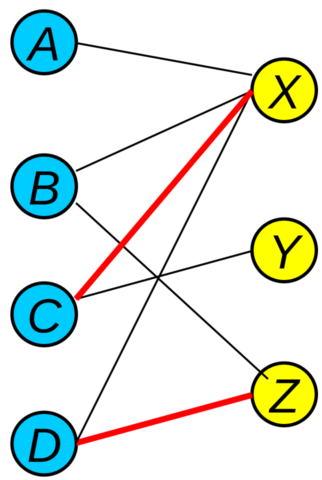
Maximum bipartite matching

- Given an undirected graph G=(V,E), a matching is a subset of edges M of E such that for all vertices v in V, at most one edge of M is incident on v
- We say the vertex v is matched by the matching M if some edge in M is incident on V, otherwise v is unmatched
- A maximum matching is a matching of maximum cardinality that is a matching M is such that for any other matching M', $|M| \geq |M'|$
- Maximum bipartite matching is the maximum matching of an undirected graph G=(V,E), where V is partitioned into disjoint sets L and R and all edges in E go between L and R

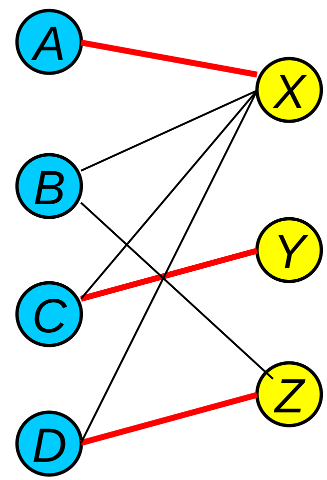
Model for Matching Problem

- Men on leftmost set, women on rightmost set, edges if they are compatible





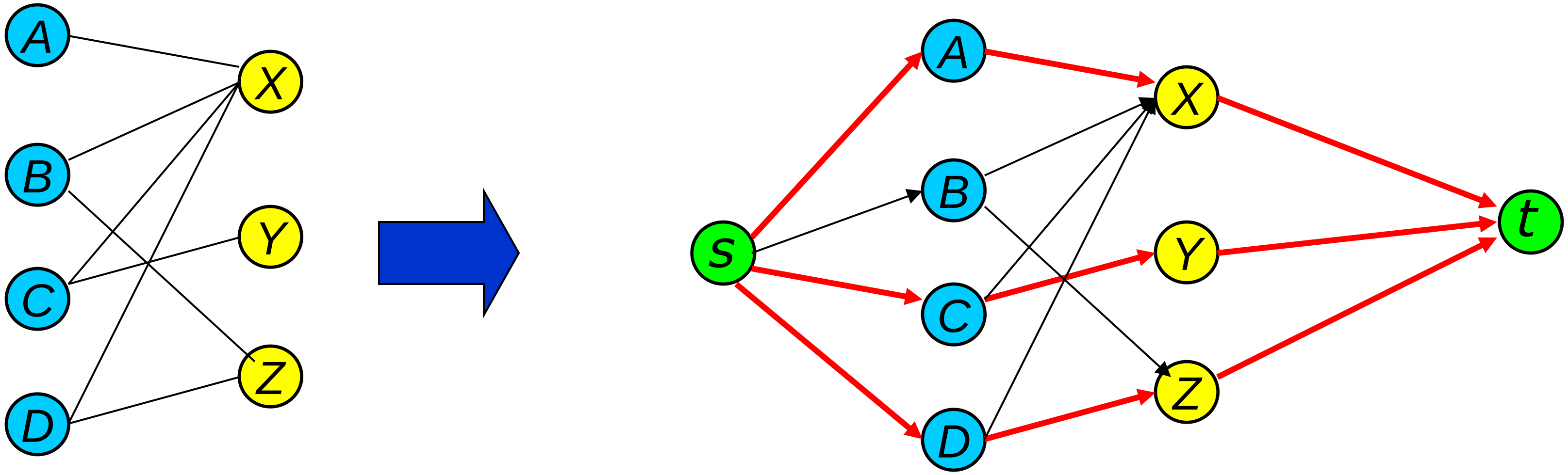
A matching



Maximum matching

Solution Using Max Flow

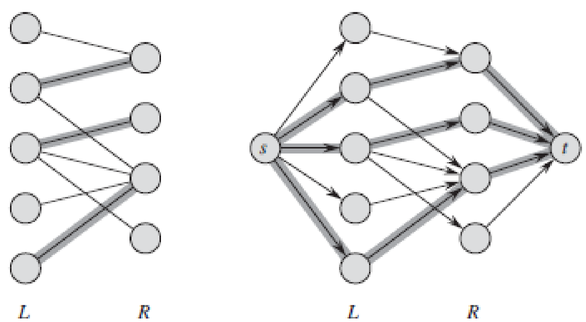
- Add a supersouce, supersink, make each undirected edge directed with a capacity of 1



Since the flow of each edge 1, flow conservation prevents multiple matchings

Running time of Bipartite matching

- For a bipartite G=(V,E), the corresponding flow network is G'=(V',E')
  - V'=V $\cup$ {s,t}
  - E' is $E' = \{(s,u) : u \in L\} \cup \{(u,v) : (u,v) \in E\} \cup \{(v,t) : v \in R\}$.



  - Since each vertex in V has at least one incident edge, $|E| \geq |V|/2$
    - $|E| \leq |E'| = |E|+|V| \leq 3|E|$

Maximum matching of the bipartite graph is same as maximum flow of the G'

Solution Using Max Flow

- Any matching in a bipartite graph has cardinality at most min(|L|,|R|)=O(|V|)
- Since capacity of each edge is 1, he value of the maximum flow is O(|V|)
  - Ford Fulkerson method at most O(|V|) iterations
- Each iteration of Ford Fulkerson method runs in O(|E'|) time
- Therefore, we can find maximum matching in a bipartite graph in O(|V||E'|) time
  - Since |E'| is at most 3 times |E|, the running time is O(|V||E|)