



# COMP 482 / ELEC 420

Maximum Network Flow

# Reading

- Skim [CLRS] 22-25
  - Graphs, minimum spanning trees, shortest paths
  - Covered in previous courses, but a refresher is always good.
  - See how Dijkstra's shortest path alg. can use Fibonacci heaps.
- Read [CLRS] 26.

# Flow networks

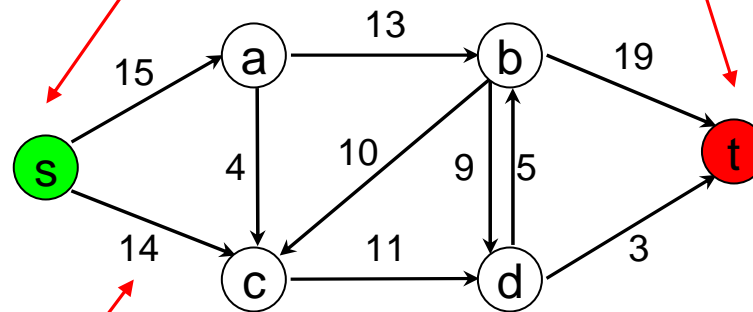
What if weights in a graph are maximum capacities of some flow of material?

- Pipe network to transport fluid/gas (e.g., water, oil, natural gas, CO<sub>2</sub>)
  - Edges – pipes
  - Vertices – junctions of pipes
- Data communication network
  - Edges – network connections of different capacity
  - Vertices – routers (do not produce or consume data just move it)
- Also used in resource planning, economics, ecosystem network analysis

# Formalization: Flow network

Directed graph  $G=(V,E)$

One *source* and one *sink* – We'll generalize later.



Each edge has  
*capacity*  $c(u,v) \geq 0$ .

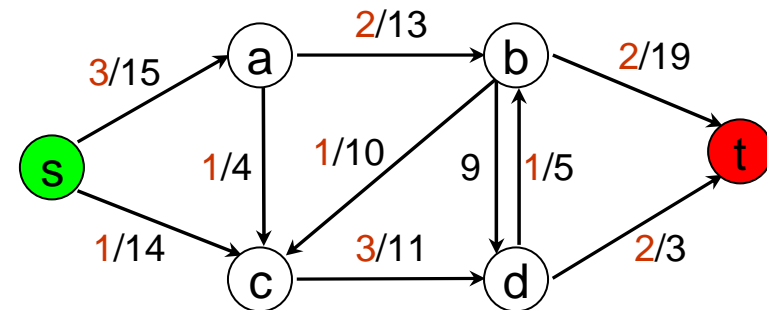
Each vertex is on some  
path from  $s$  to  $t$ .

# Formalization: Flow

How much is currently flowing –  $f : V \times V \rightarrow \mathbb{R}$

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

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



Must satisfy 3 properties:

- Capacity constraint:

$$\forall u, v \in V: f(u, v) \leq c(u, v)$$

- Skew symmetry:

$$\forall u, v \in V: f(u, v) = -f(v, u)$$

- Flow conservation:

$$\forall u \in V - \{s, t\}: f(u, V) = f(V, u) = 0$$

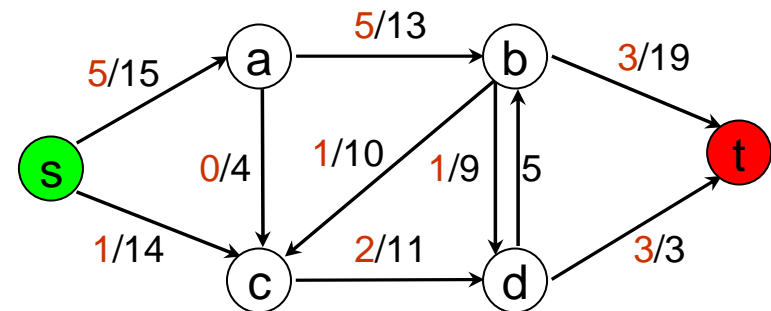
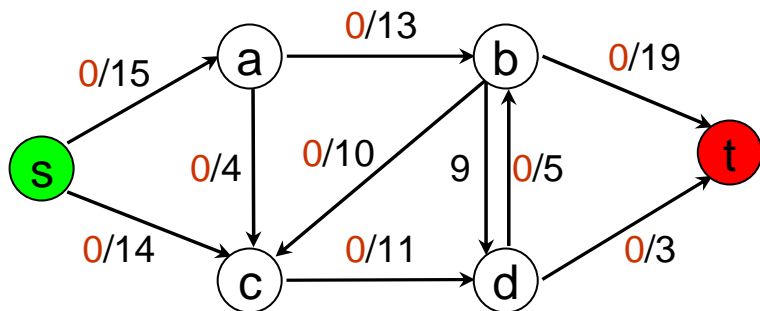
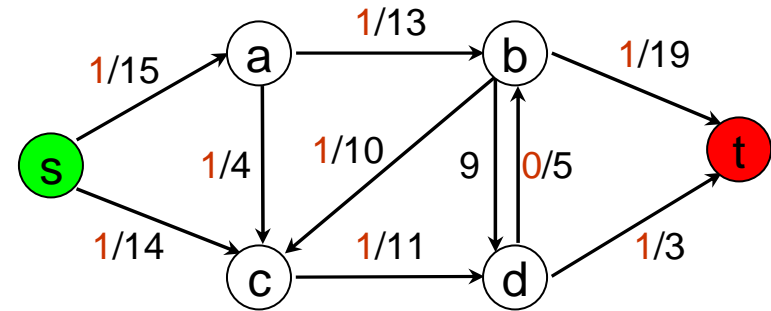
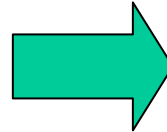
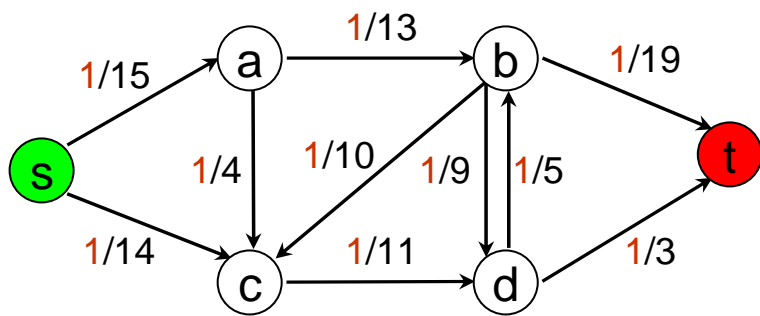
What goes in must go out.

Total value of flow  $f$ :

$$|f| = f(s, V) = f(V, t)$$

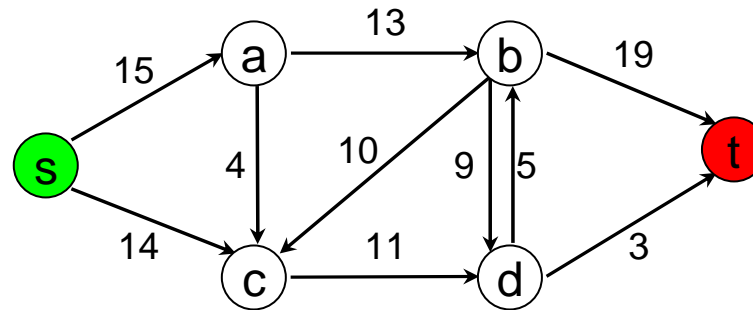
# Example flows

Valid or invalid? Why?



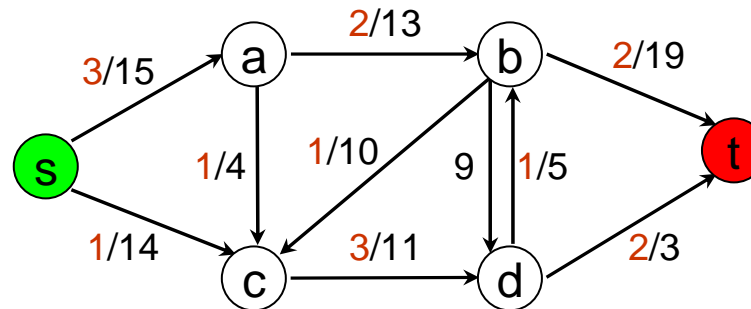
# Maximum flow problem

Find  $f$  to maximize  $|f|$ .



# Maximum flow: Algorithm idea

- If we have some flow, ...



- ...and can find an *augmenting path*  $p$ 
  - from  $s$  to  $t$
  - can add a constant amount of flow along path:  
 $\exists a > 0, \forall (u,v) \in p, f(u,v) + a \leq c(u,v)$
- Then just do it, to get a better flow!

Find an augmenting path in the example!

# Ford-Fulkerson method

**Ford-Fulkerson**( $G, s, t$ )

```
1 initialize flow  $f$  to 0 everywhere
2 while there is an augmenting path  $p$  do
3     augment flow  $f$  along  $p$ 
4 return  $f$ 
```

- How do we find/choose an augmenting path?
- How much additional flow can we send through that path?
- Does the algorithm always find the maximum flow?

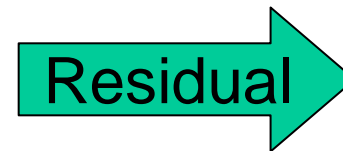
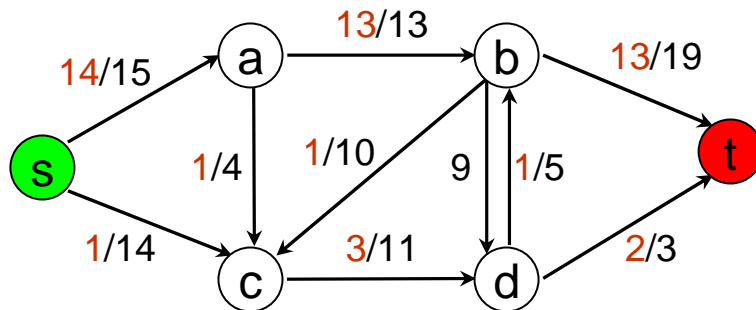


<http://algoviz.cs.vt.edu/AlgovizWiki/NetworkFlow>

# Augmenting

Augmenting path – any path in the *residual network*:

- Residual network:  $G_f = (V, E_f)$   
 $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$
- Residual capacities:  $c_f(u, v) = c(u, v) - f(u, v)$
- Residual capacity of path  $p$ :  $c_f(p) = \min_{(u, v) \in p} c_f(u, v)$



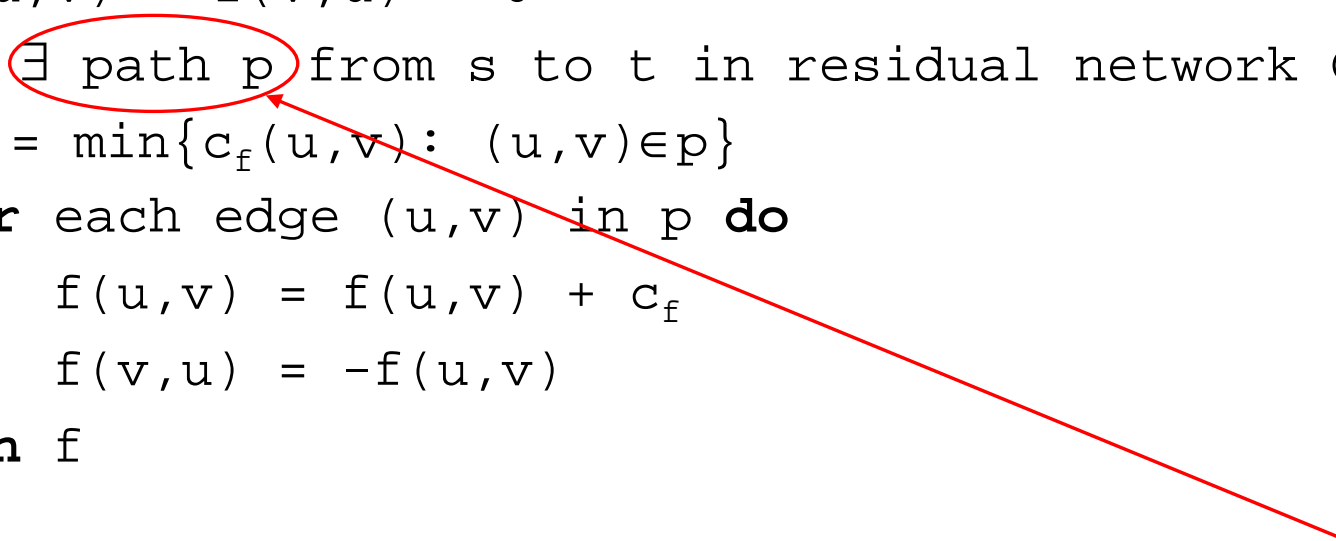
Residual capacity  
of path (s,c,d,t)?

Observe – Edges in  $E_f$  are either edges in  $E$  or their reversals:  $|E_f| \leq 2|E|$ .

# Ford-Fulkerson method, with details

**Ford-Fulkerson**( $G, s, t$ )

```
1  for each edge  $(u,v) \in G.E$  do
2       $f(u,v) = f(v,u) = 0$ 
3  while  $\exists$  path  $p$  from  $s$  to  $t$  in residual network  $G_f$  do
4       $c_f = \min\{c_f(u,v) : (u,v) \in p\}$ 
5      for each edge  $(u,v)$  in  $p$  do
6           $f(u,v) = f(u,v) + c_f$ 
7           $f(v,u) = -f(u,v)$ 
8  return  $f$ 
```



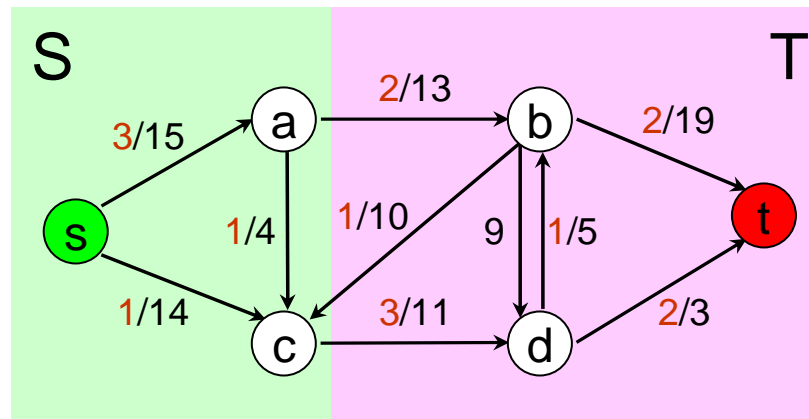
Algorithms based on this method differ in how they choose  $p$ .

# Does it always find a maximum flow?

First, some definitions:

*Cut*

– a partition of  $V$  into  $S, T$  such that  $s \in S, t \in T$



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

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

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

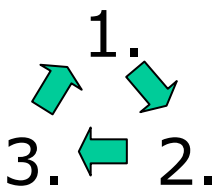
*Minimum cut* – a cut with the smallest capacity of all cuts

# Does it always find a maximum flow?

Max-flow min-cut theorem:

The following are equivalent statements:

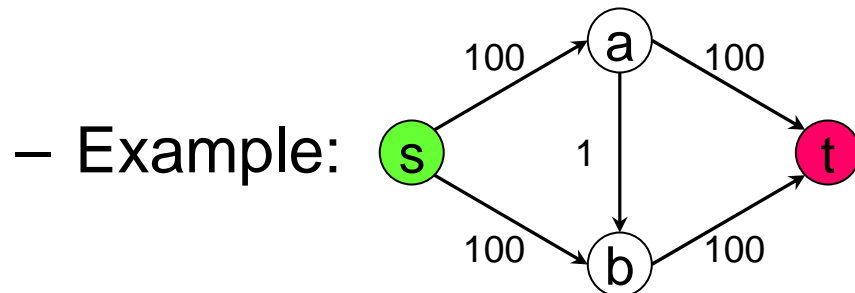
1.  $f$  is a maximum flow in  $G$ .
2. The residual network  $G_f$  contains no augmenting paths.
3.  $|f| = c(S,T)$ , for some cut  $(S,T)$  of  $G$ .

We prove three parts: 

From this we have  $2. \rightarrow 1.$ , which means that the Ford-Fulkerson method always correctly finds a maximum flow.

# What is the worst-case running time?

- Augmentation = ?  $O(E)$
- How many augmentations?
  - Let's assume integer flows.
  - Each increases the value of the flow by some integer.
  - $O(|f^*|)$ , where  $f^*$  is the max-flow.
- Total *worst-case* =  $O(E \times |f^*|)$ .

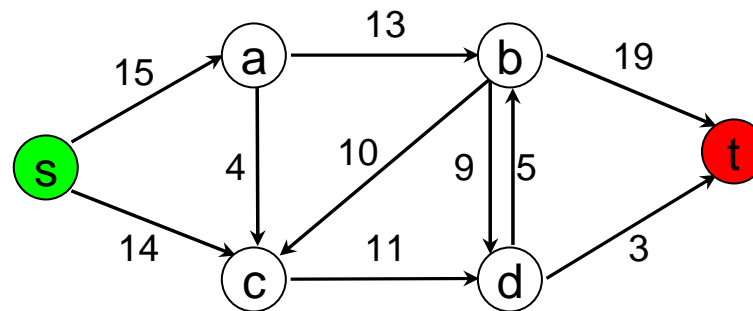


- How an augmenting path is chosen is very important!

# Edmonds-Karp algorithm

Use **shortest** augmenting path (in #edges).

Run algorithm on our example:



# Edmonds-Karp algorithm analysis: 1

Augmentation =  $O(E)$  – Breadth-first search

Will prove: #augmentations =  $O(VE)$ .

Let  $d(v)$  be distance from  $s$  to  $v$  in residual network.

Will prove: Every  $|E|$  iterations,  $d(t)$  increases by  $\geq 1$ .

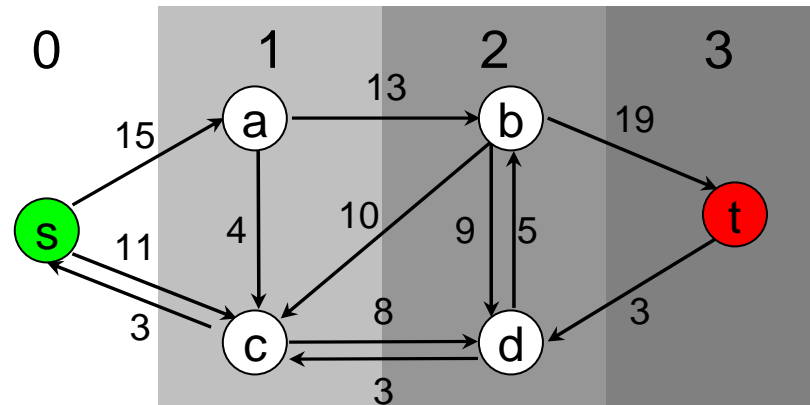
$d(t)$  can increase at most  $|V|$  times  $\rightarrow O(VE)$  iterations

Total =  $O(VE^2)$

# Edmonds-Karp algorithm analysis: 2

Will prove: Every  $|E|$  iterations,  $d(t)$  increases by  $\geq 1$ .

Consider the residual network in levels according to  $d(v)$ :



As long as  $d(t)$  doesn't change, the paths found will only use forward edges.

- Each iteration saturates & removes at least 1 forward edge, and adds only backward edges (so no distance ever drops).
- After removing  $|E| - d(t) + 1$  forward edges,  $t$  will be disconnected from  $s$ .

So, within  $|E|$  iterations, either

- $t$  is disconnected, & algorithm terminates, or
- A non-forward edge used, &  $d(t)$  increased.

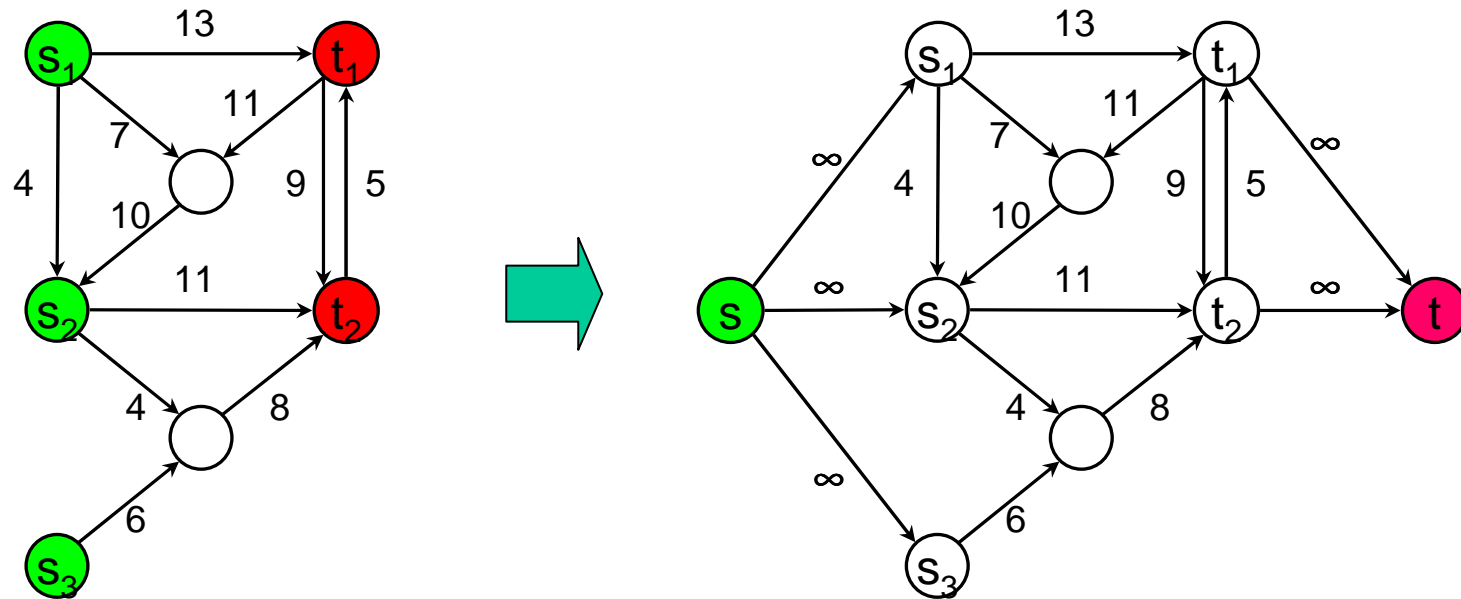
# Other Max-Flow Algorithms

More complex, but asymptotically faster.  
Problem important, thus well-studied.

- Push-relabel – Goldberg & Tarjan (1986)
  - $O(VE \log (V^2/E + 2))$
  - Described in [CLRS].
  - In practice, most-used approach.
- King, Rao, & Tarjan (1994)
  - $O(VE \log_{E/(V \log V)} V)$
- Goldberg & Rao (1998)
  - $O(\min(V^{2/3}, E^{1/2}) E \log(V^2/E+2) \log (\max_{(u,v) \in E} c(u,v)))$

# Variation: Multiple sources or sinks

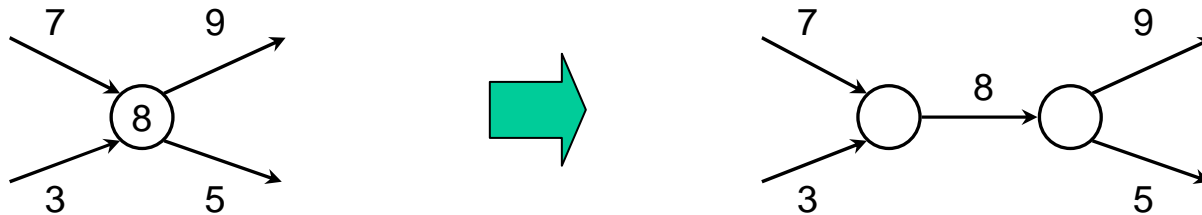
What if we have more sources or sinks?



Augment the graph to make it with one source and one sink!

## Variation: Node capacities

What if nodes have maximum flows?



Convert into a edge capacity.

# Variations

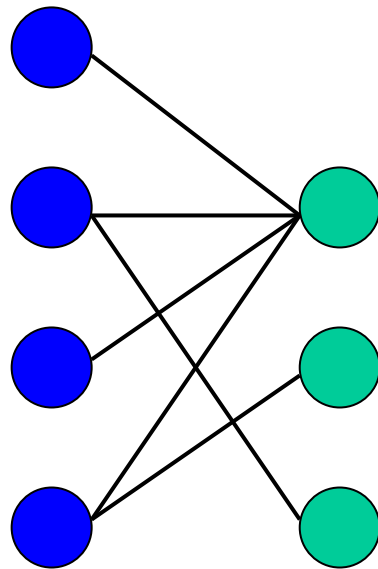
Any combination of

- Multiple sources & sinks
- Node capacities (& possibly no edge capacities)
- Minimum edge/node flows
- Sources & sinks also conserve flow – *circulation*
- Multiple distinct flow *commodities*
- Flows have costs, which are to be minimized
- Integral flows

Some can be restated as max-flow, some can be solved by extensions of same algs., some NP-complete

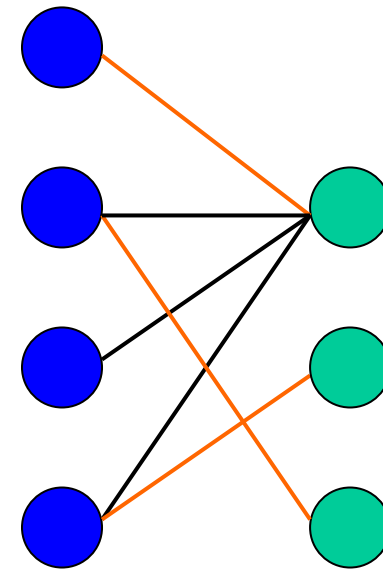
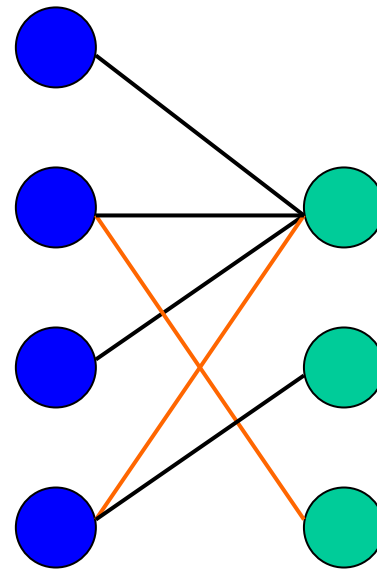
# Application: Maximum Bipartite Matching

**Matching** in a graph is a subset  $M$  of edges such that each vertex has at most one edge of  $M$  incident on it. It puts vertices in pairs.



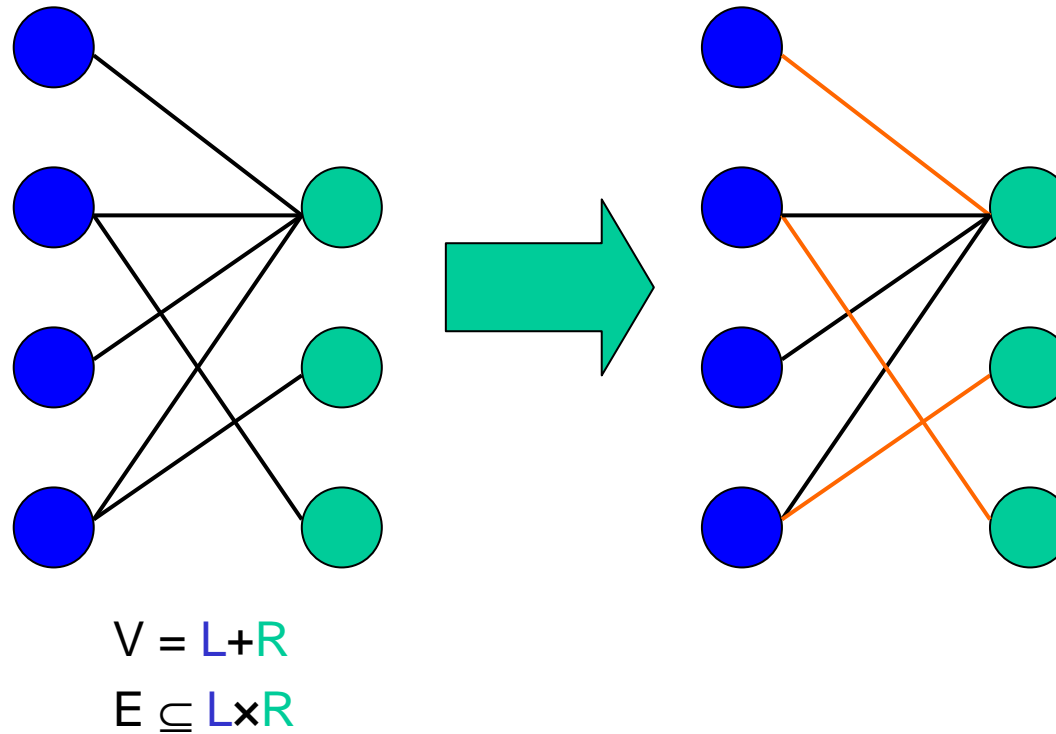
Bipartite:  $V = L+R$   
 $E \subseteq L \times R$

E.g., dating agency



A maximum

# Application: Maximum Bipartite Matching



- How can we reformulate as a max-flow problem?
- What is the running time, using Edmonds-Karp?