Directed Graphs

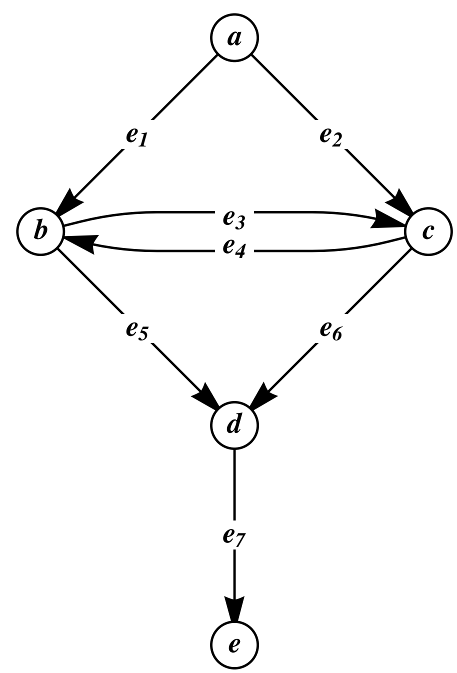
Definition: A **directed graph** (or **digraph**) is a graph $G = (V, E)$, where each edge $e = vw$ is directed from one vertex to another:

$$e : v \longrightarrow w \qquad \text{or} \qquad e : w \longrightarrow v.$$

Remark: The edge $e : v \longrightarrow w$ is different from $e' : w \longrightarrow v$ and a digraph including both is not considered to have multiple edges.

Definition: The **in-degree** of a vertex v is the number of edges directed *toward* v.

Definition: The **out-degree** of a vertex v is the number of edges directed *away from* v.



Definition: A **source** s is a vertex with in-degree 0.
Definition: A **sink** t is a vertex with out-degree 0.

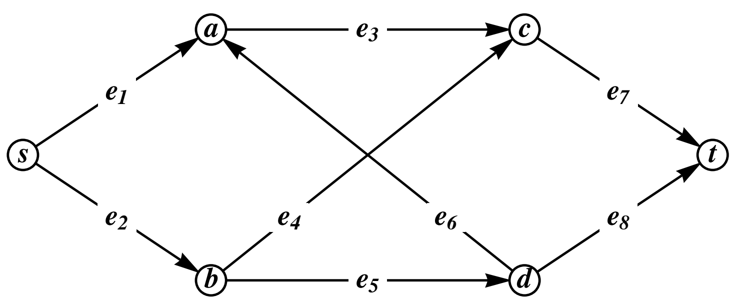
Important: Any **path** or **cycle** in a digraph must respect the direction on each edge.

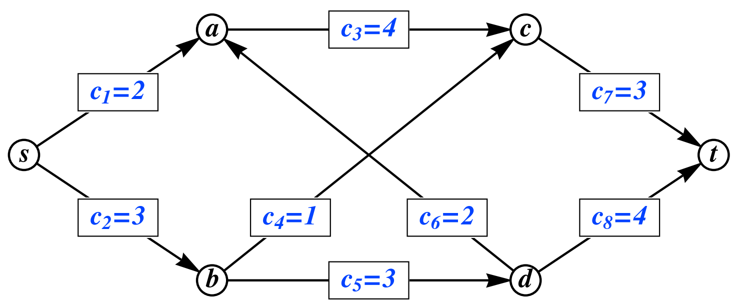
Network Flows

Definition: A **network** is a directed graph with additional structure:

- There are two distinguished vertices, s (a source) and t (a sink).
- Each edge e has a **capacity** c_e. [*Some sort of limit on flow.*]

Idea: Graph networks represent real-world networks such as traffic, water, communication, etc.



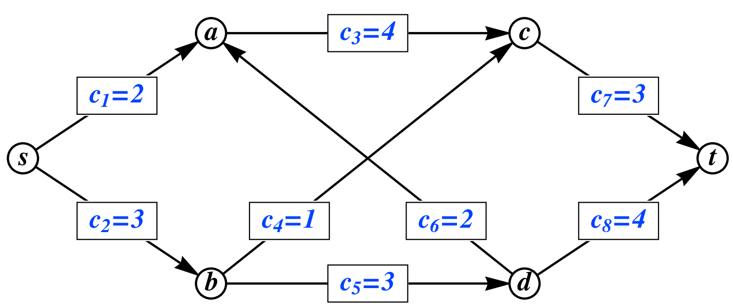


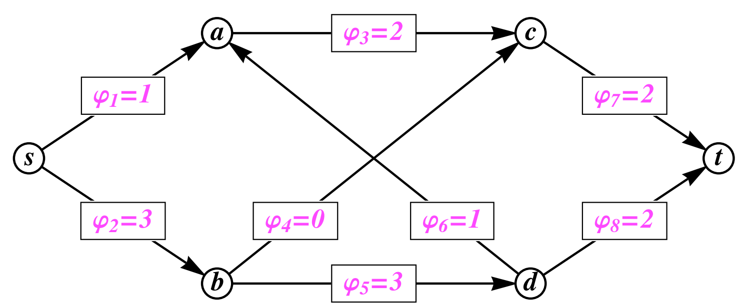
Goal: Send as much "stuff" from s to t while respecting capacities.

Network Flows

Definition: Given a network G, a **flow** $\vec{\varphi} = \{\varphi_e\}_{e \in E(G)}$ on G is an assignment of values φ_e to every edge of G satisfying:

- $\boxed{0 \le \varphi_e \le c_e}$ for every edge $e \in E(G)$.
  - *The flow respects the capacities.*
- $\boxed{\displaystyle\sum_{e \text{ into } v} \varphi_e = \sum_{e \text{ out of } v} \varphi_e}$ for every vertex $v \in V(G)$ except s or t.
  - *Obeys "conservation of flow" except at s and t.*





Definition: When $\varphi_e = c_e$, we say that e is **saturated**, or **at capacity**.

Maximum Flow

Theorem: Given a flow $\vec{\varphi}$ on a network G, the net flow out of s is equal to the net flow into t. Symbolically, $\sum_{e \text{ out of } s} \varphi_e = \sum_{e \text{ into } t} \varphi_e$.

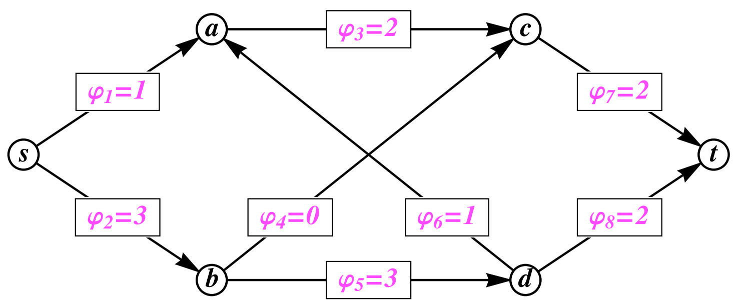
Proof: Create a new network G' by adding to G an edge $e_\infty : t \rightarrow s$ with infinite capacity, and place flow

$$\varphi_\infty = \sum_{e \text{ out of } s} \varphi_e$$

on e_∞. In G', flow is now conserved at every vertex except possibly t. By Kirchhoff's Global Current Law (Theorem 6.2.2), flow must be conserved at t as well.

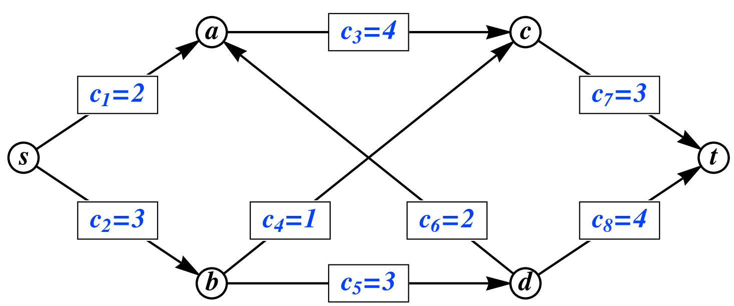
Maximum Flow

Definition: The **throughput** or **value** of a flow $\vec{\varphi}$ is $\sum_{e \text{ out of } s} \varphi_e$, denoted $|\vec{\varphi}|$.



Idea: The throughput is the amount of "stuff" flowing through G.

In our example, $|\vec{\varphi}| =$ __________________.



Goal: For a given network, find the flow with the largest throughput.

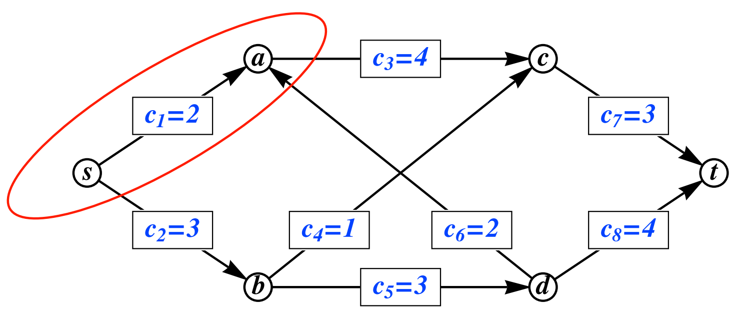
This problem is called **maximum flow**.

MAX FLOW: $$\underset{\text{over all flows } \vec{\varphi} \text{ on } G}{\text{maximize}} \quad |\vec{\varphi}|$$

st-Cuts

A related problem in network theory has to do with *st*-cuts.

Definition: Let G be a network. Let X be a set of vertices containing s and not containing t. An *st*-**cut** $[X, X^c]$ is the **set of edges** between a vertex in X and a vertex in X^c (in either direction).



$X =$

$X^c =$

$[X, X^c] =$

$|[X, X^c]| =$

Definition: The **capacity** of an *st*-cut, denoted $|[X, X^c]|$ is the sum of the capacities of the edges **from** a vertex in X **to** a vertex in X^c.

Idea: The capacity of a cut is the limit for how much "stuff" can go from X to X^c.

⋆ Do **not** subtract the capacities of the edges going the other way. ⋆

Max Flow / Min Cut

Goal: For a given network, find the *st*-cut with the smallest capacity.

This problem is called **minimum cut**.

MIN CUT: $$\underset{\text{over all cuts } [X, X^c] \text{ on } G}{\text{minimize}} \quad |[X, X^c]|$$

The problems Max Flow and Min Cut are related because for any flow $\vec{\varphi}$, the net flow through the edges of any *st*-cut $[X, X^c]$ is at most the capacity of $[X, X^c]$. This proves:

Theorem: For any flow $\vec{\varphi}$ and *st*-cut $[X, X^c]$, $|\vec{\varphi}| \leq |[X, X^c]|$.

Theorem: For any maximum flow $\vec{\varphi}^*$ and minimum *st*-cut $[X^*, X^{*c}]$,

$$|\vec{\varphi}^*| \leq |[X^*, X^{*c}]|.$$

So if there exists a flow $\vec{\varphi}$ and *st*-cut $[X^*, X^{*c}]$ where equality holds, then $\vec{\varphi}$ is a maximum flow and $[X^*, X^{*c}]$ is a minimum cut

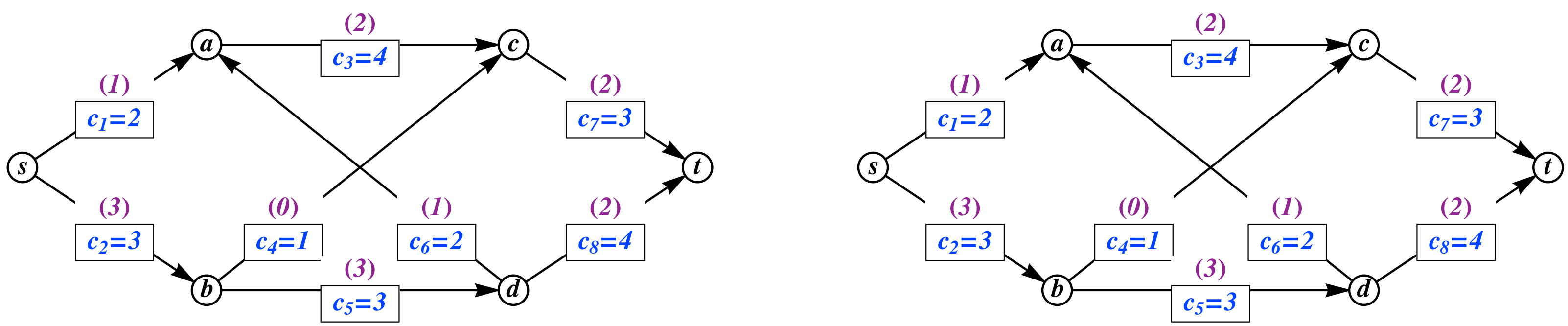
Max Flow / Min Cut Theorem

Theorem: (Ford, Fulkerson, 1955) In any network G, the value of any maximum flow is equal to the capacity of any minimum cut.

Proof: Use the Ford–Fulkerson Algorithm to find a max flow.

Idea: Similar to the Hungarian Algorithm for finding a max matching, we will augment an existing flow $\vec{\varphi}$.

Question: What does it look like to augment *a flow*?



We can augment in the forward direction when ________________.
We can augment in the backward direction when ________________.

We'll create a *companion graph* to keep track of augmenting paths.

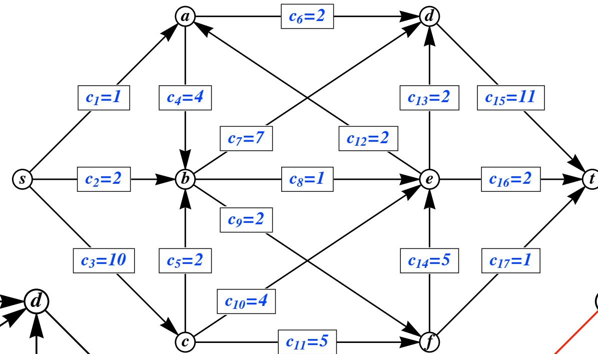
Max Flow / Min Cut Theorem

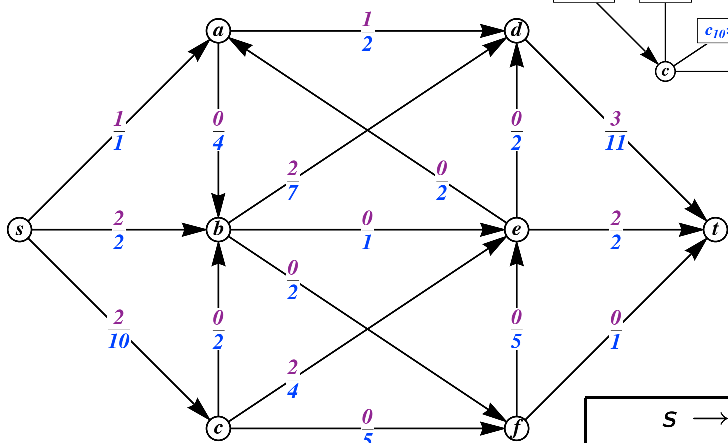
Theorem: (Ford, Fulkerson, 1955) In any network G, the value of any maximum flow is equal to the capacity of any minimum cut.

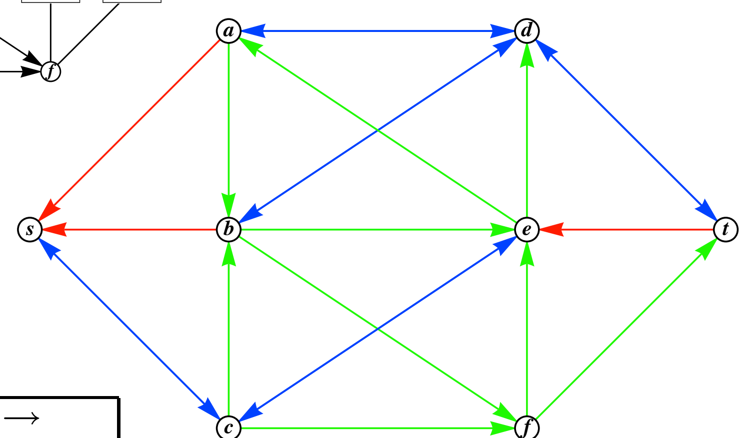
Proof: Use the **Ford–Fulkerson Algorithm**, which finds a max flow.

1. Start with any flow $\vec{\varphi}$ on G.
2. Draw the **flow companion graph** using the underlying graph
   - If $\varphi_e = 0$, orient the edge e forward only.
   - If $0 < \varphi_e < c_e$, orient the edge e both forward and backward.
   - $\varphi_e = c_e$, orient the edge e backward only.
3. ⋆ If there is an st-path in the flow companion graph, send as many units of flow as possible through this path. Repeat Step 2.
   ⋆ If there is no st-path in the flow companion graph, STOP.
   $\longrightarrow$ The current flow is a maximum flow. $\longleftarrow$
   In addition, let X be the set of vertices reachable from s in the flow companion graph. Then $[X, X^c]$ is a minimum st-cut.

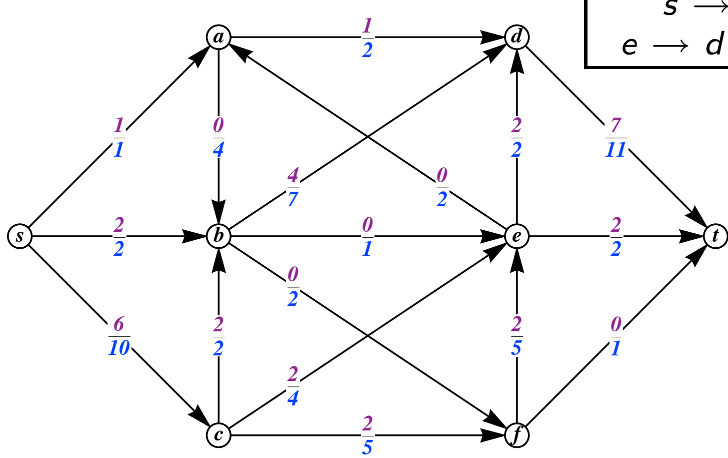
A Ford–Fulkerson Algorithm Example

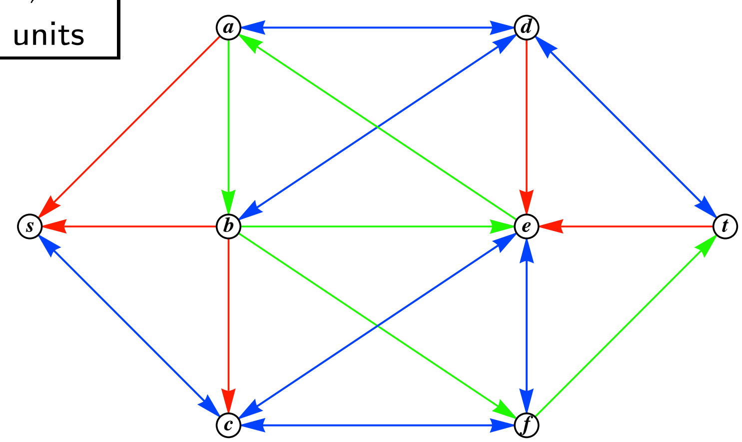






$s \longrightarrow c \longrightarrow b \longrightarrow d \longrightarrow t$: 2 units
$s \longrightarrow c \longrightarrow f \longrightarrow e \longrightarrow d \longrightarrow t$: 2 units

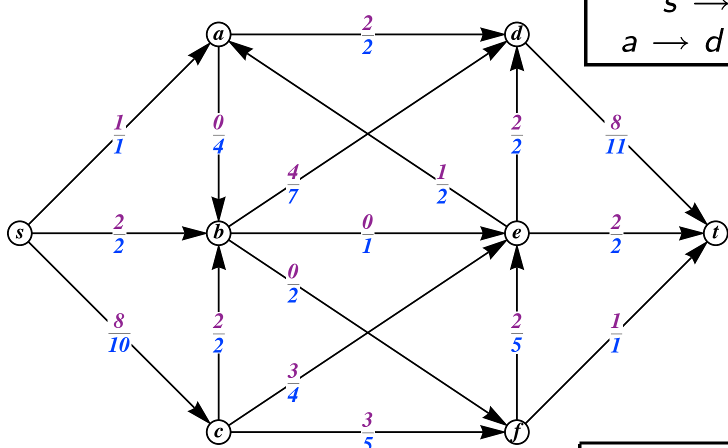


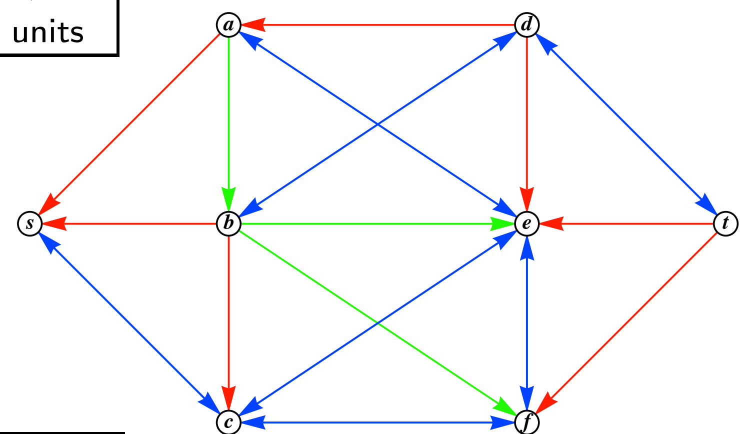


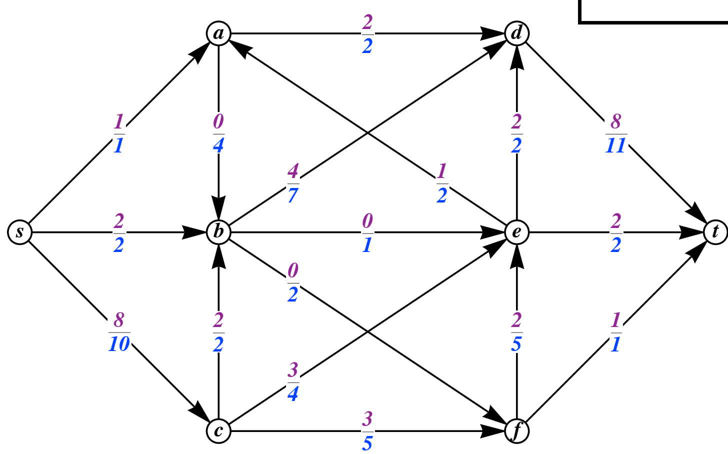
A Ford–Fulkerson Algorithm Example

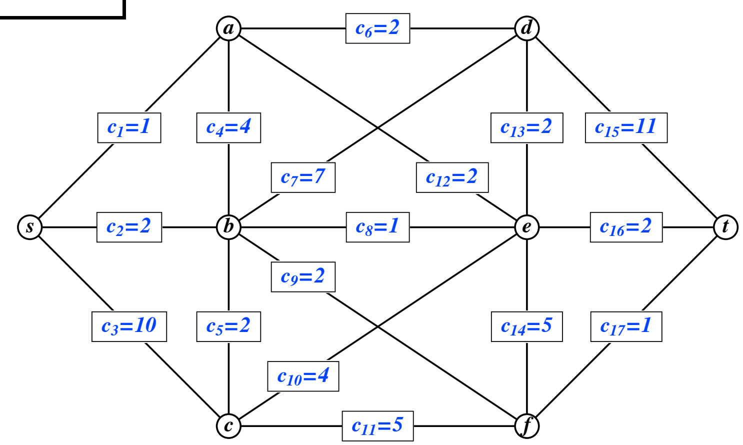
$s \longrightarrow c \longrightarrow f \longrightarrow t$: 1 unit

$s \longrightarrow c \longrightarrow e \longrightarrow a \longrightarrow d \longrightarrow t$: 2 units









$X = \{\underline{\hspace{3cm}}\}$, $[X, X^c] = \{\underline{\hspace{5cm}}\}$, and $|[X, X^c]| = \underline{\hspace{2cm}}$.