

Reminder: 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:

$$e : v \rightarrow w \quad \text{or} \quad e : w \rightarrow v.$$

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

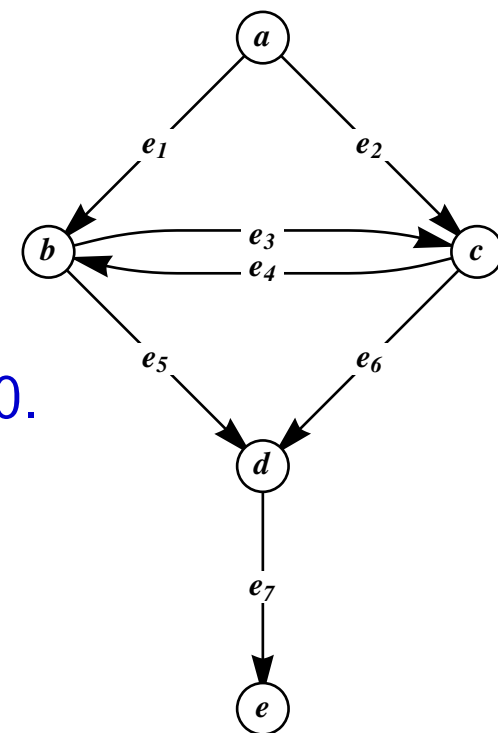
Definition. The **in-degree** of a vertex v is the number of edges directed *toward* v .

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.

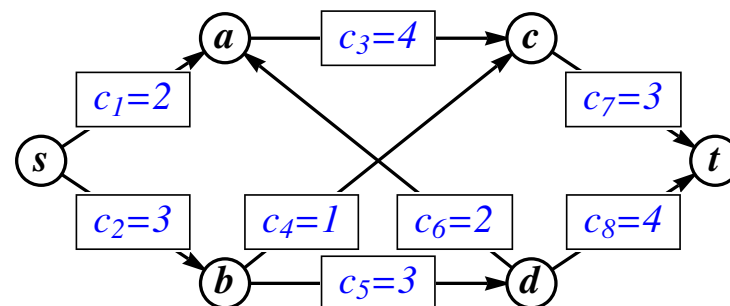
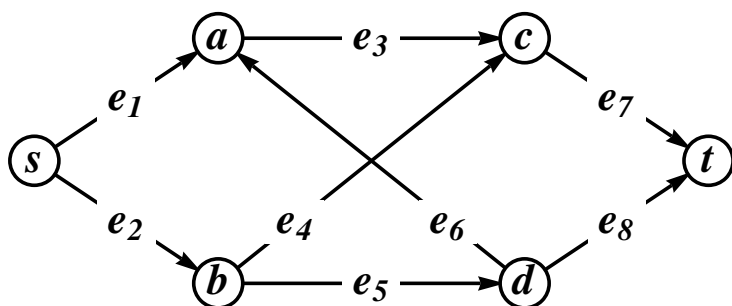
Important. Any **path** or **cycle** in a digraph must respect the direction on each edge.



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

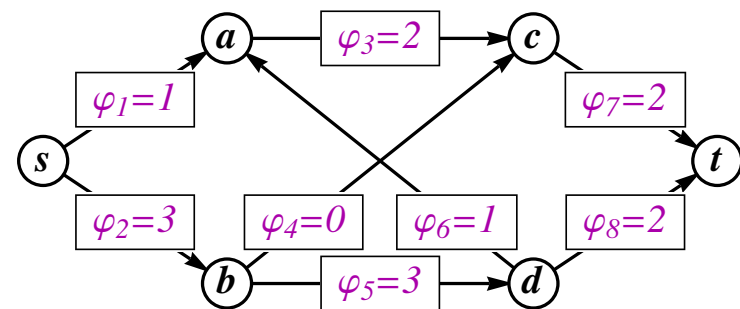
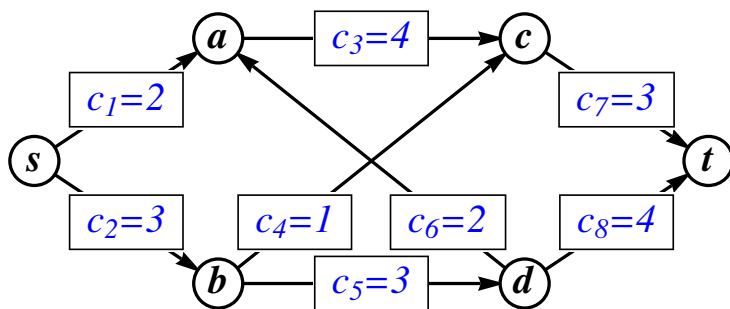
Definition. Given a network G , a **flow** $\vec{\varphi} = \{\varphi_e\}_{e \in E(G)}$ on G is an assignment of values φ_e to every edge of G satisfying:

▶ $0 \leq \varphi_e \leq c_e$ for every edge $e \in E(G)$.

▶ *The flow respects the capacities.*

▶ $\sum_{e \text{ into } v} \varphi_e = \sum_{e \text{ out of } v} \varphi_e$ 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

$$\varphi_\infty = \sum_{e \text{ out of } s} \varphi_e \quad \text{on } e_\infty.$$

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

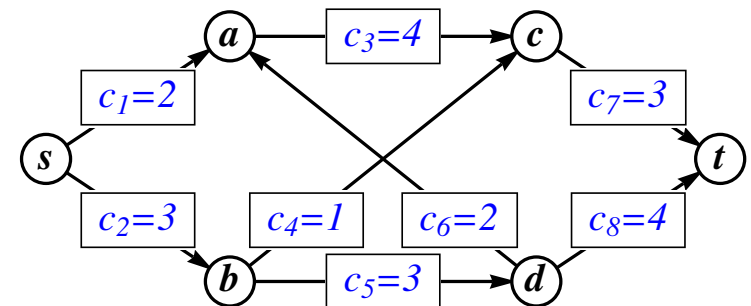
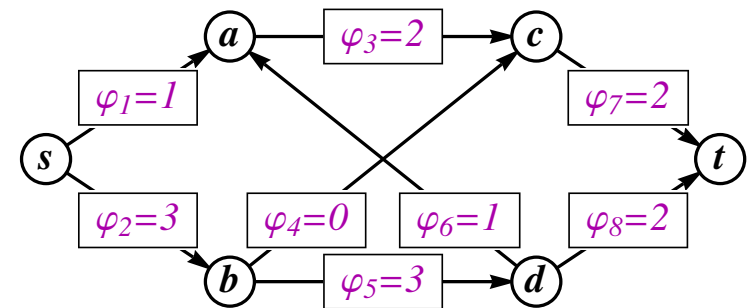
Definition. The **throughput** or **value** of a flow $\vec{\varphi}$ is $\sum_{e \text{ out of } s} \varphi_e$, denoted $|\vec{\varphi}|$.

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

In our example, $|\vec{\varphi}| = \underline{\hspace{2cm}}$.

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

This problem is called **maximum flow**.



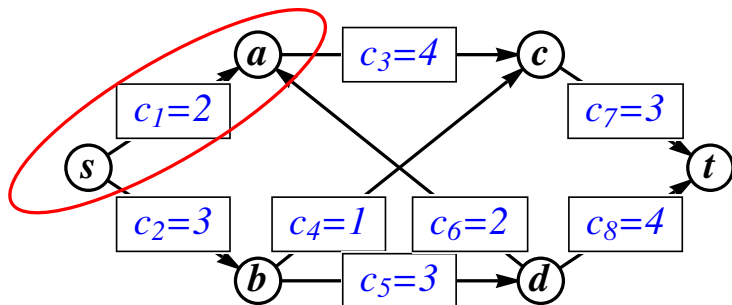
MAX FLOW

maximize
over all flows $\vec{\varphi}$ on G $|\vec{\varphi}|$

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: A cut's capacity limits how much "stuff" can go from X to X^c .

★ Capacity of $[X, X^c]$ is calculated from c_i 's, not φ_i 's. ★

★ 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**.

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

Key insight: **MAX FLOW** and **MIN CUT** are related!

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.

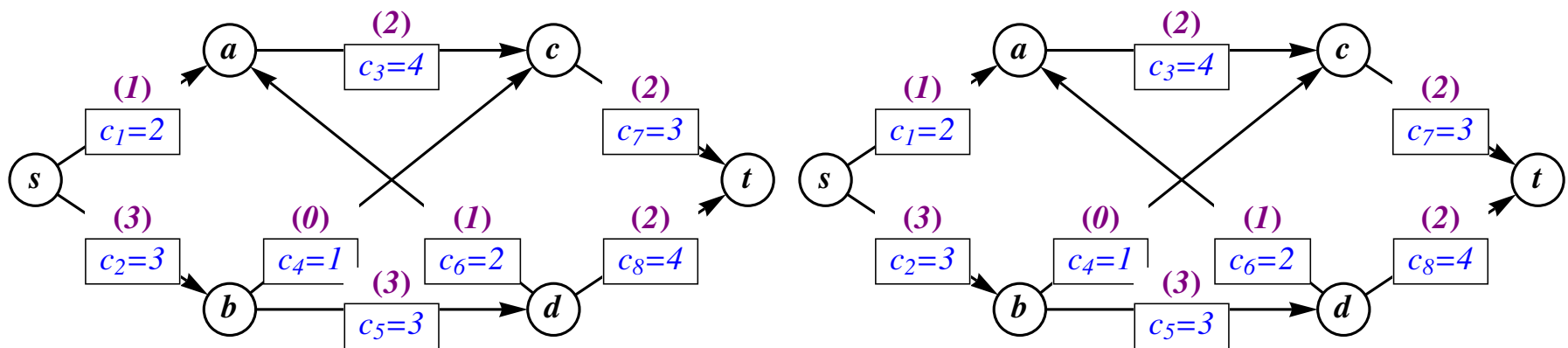
Max Flow / Min Cut Theorem

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: Add additional flow to existing flows using augmenting paths. (Similar to the Hungarian Algorithm for finding a max matching.)

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 _____.

So: Use a *companion graph* to keep track of augmentable edges/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.

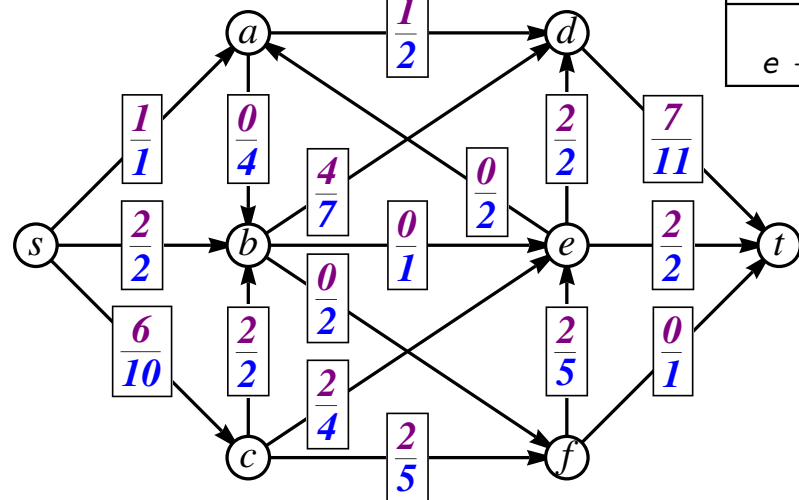
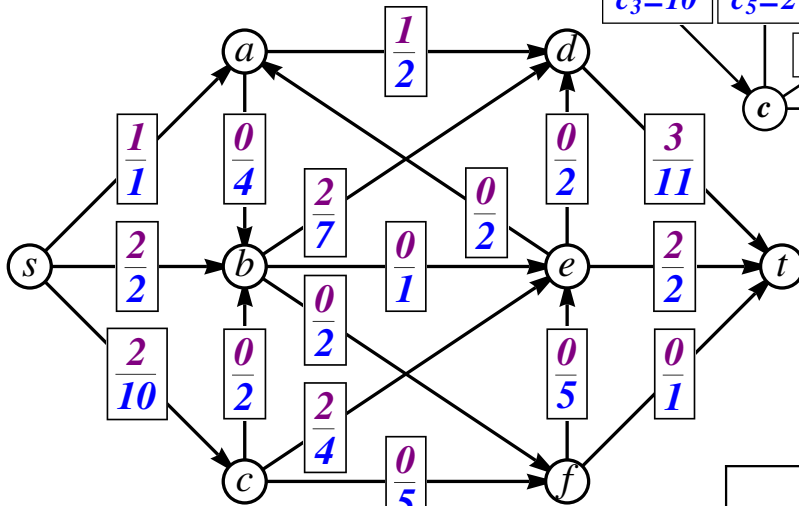
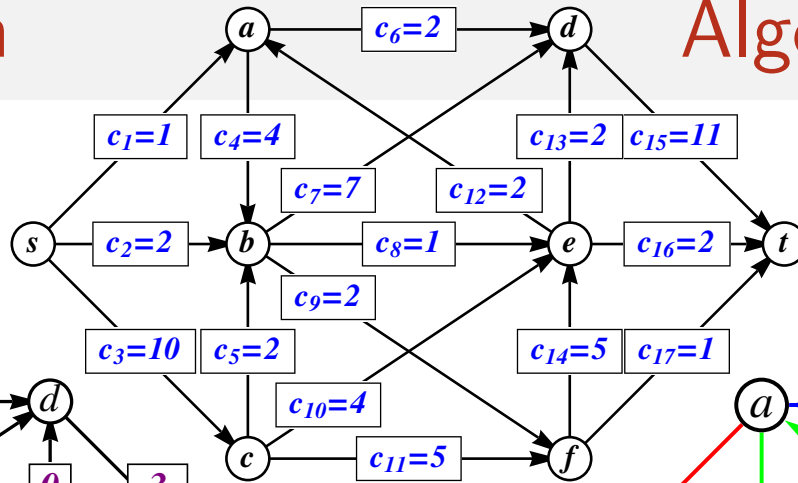
Proof. Use the **Ford–Fulkerson Algorithm**, which finds a max flow.

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.
- Upon STOP, the current flow is a maximum flow. ←

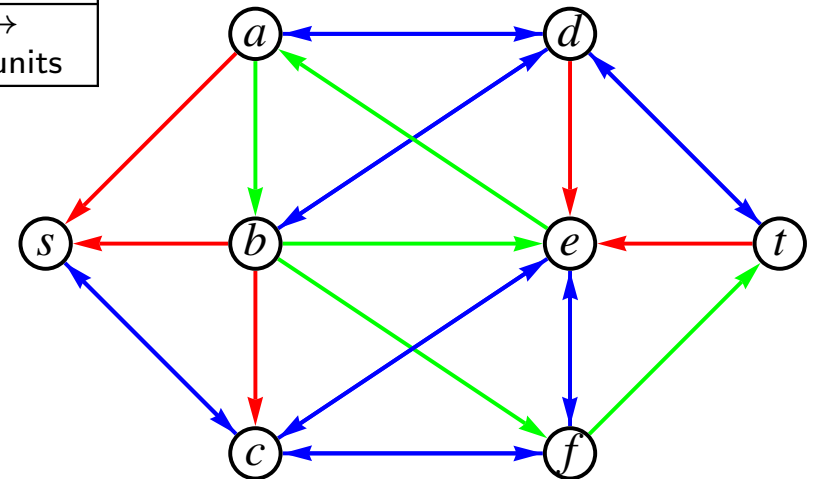
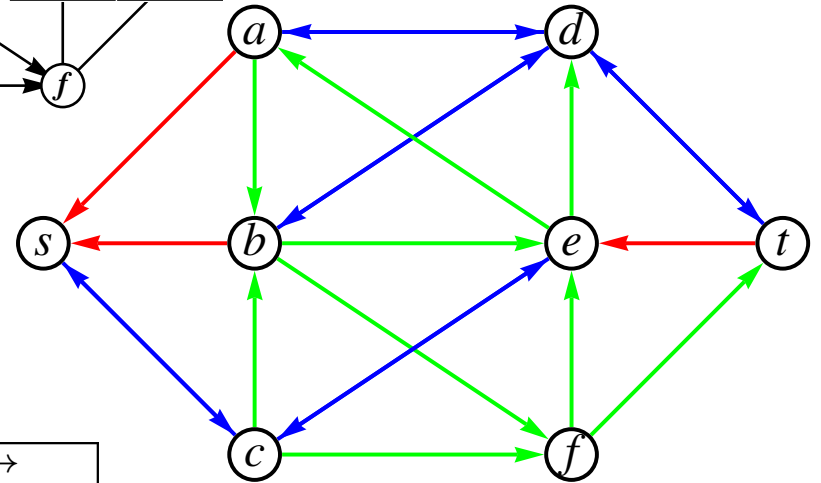
In addition, let X be the set of vertices **reachable from s** in the flow companion graph. Then $[X, X^c]$ is a minimum st -cut.

A Ford–Fulkerson

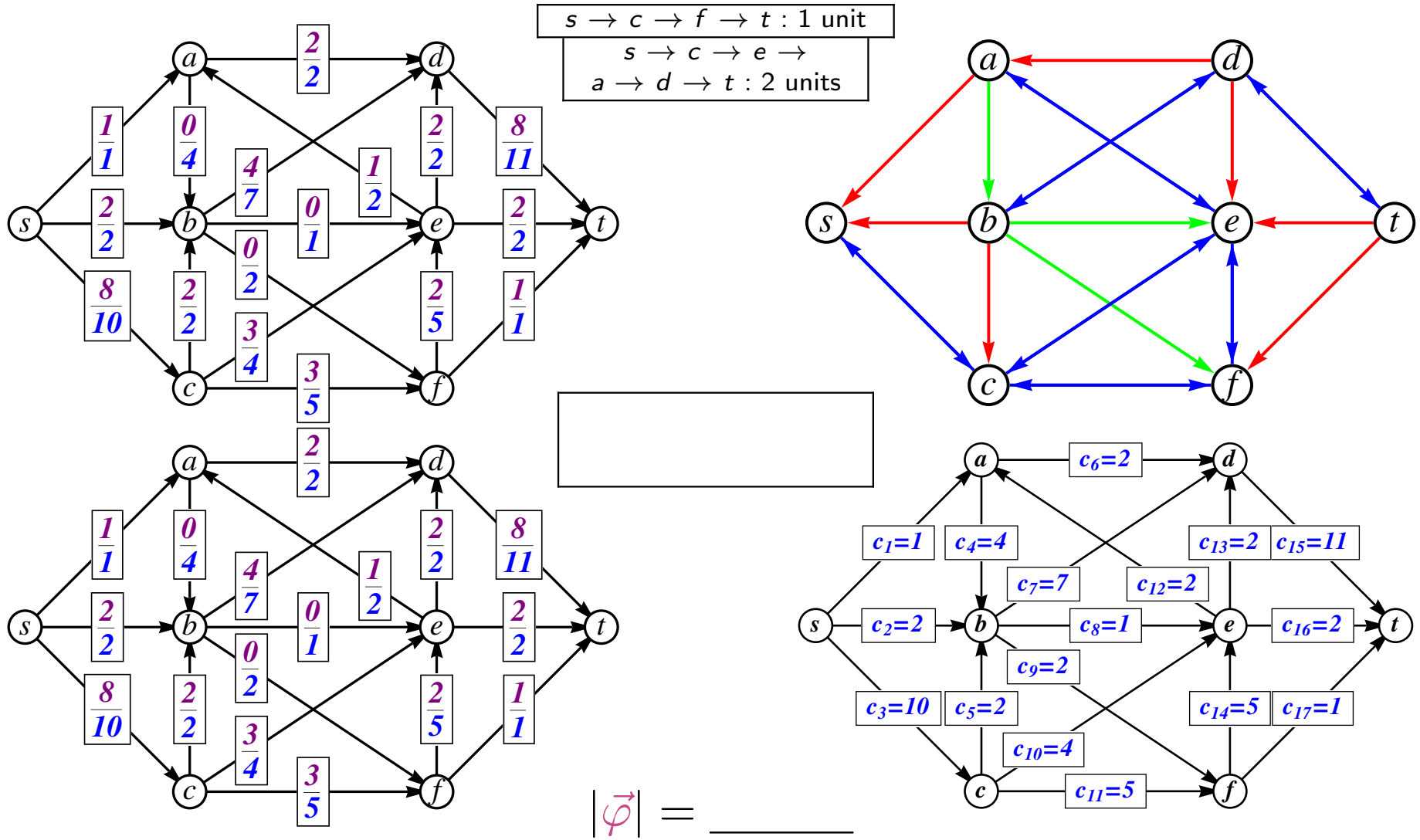
Algorithm Example



$s \rightarrow c \rightarrow b \rightarrow$
$d \rightarrow t : 2 \text{ units}$
$s \rightarrow c \rightarrow f \rightarrow$
$e \rightarrow d \rightarrow t : 2 \text{ units}$



A Ford–Fulkerson Algorithm Example



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

Correctness of the Ford–Fulkerson Algorithm

Claim. The Ford–Fulkerson Algorithm gives a maximum flow.

Proof. We must show that the algorithm always stops, and that when it stops, the output is indeed a maximum flow.

★ We will consider the case of integer capacities.

The algorithm terminates.

- ▶ Each iteration increases the throughput of the flow by an integer.
- ▶ The sum of the capacities on the edges out of s is finite.

The output is a maximum flow. Upon termination:

- ▶ There are no flow augmenting paths in the companion graph, so:
- ▶ Edges from X to X^c are full and edges from X^c to X are empty.
- ▶ The capacity of $[X, X^c]$ equals the throughput of the flow $\vec{\varphi}$.

Conclusion. The flow is a max flow and the st -cut is a min cut.

Remarks about Ford-Fulkerson

- ▶ When using the algorithm, it is important to increase the flow by as much as possible at each step.
- ▶ When the capacities are integers, we always increase the throughput by integers. The algorithm does work when the capacities are not integers, but the proof is more involved.
- ▶ As presented here, this algorithm may be very slow.

