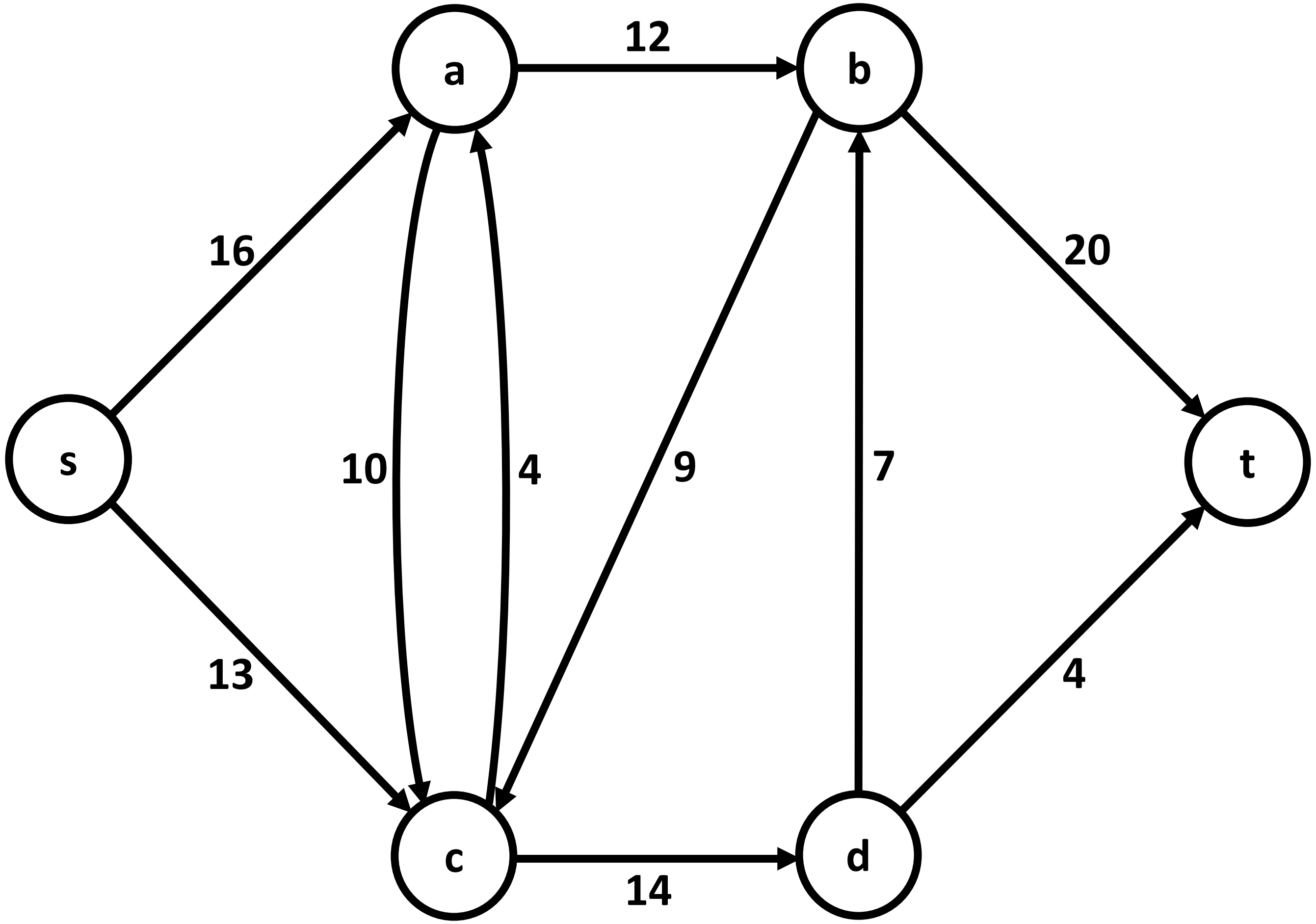
Flow Networks

Flow Network

- A flow network $G = (V, E)$ is a weighted directed graph
  - Each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$.
  - If (u, v) does not belong to E, $c(u, v) = 0$.
  - Two special vertices are considered: a source s and a sink t.

Maximum Flow Problem

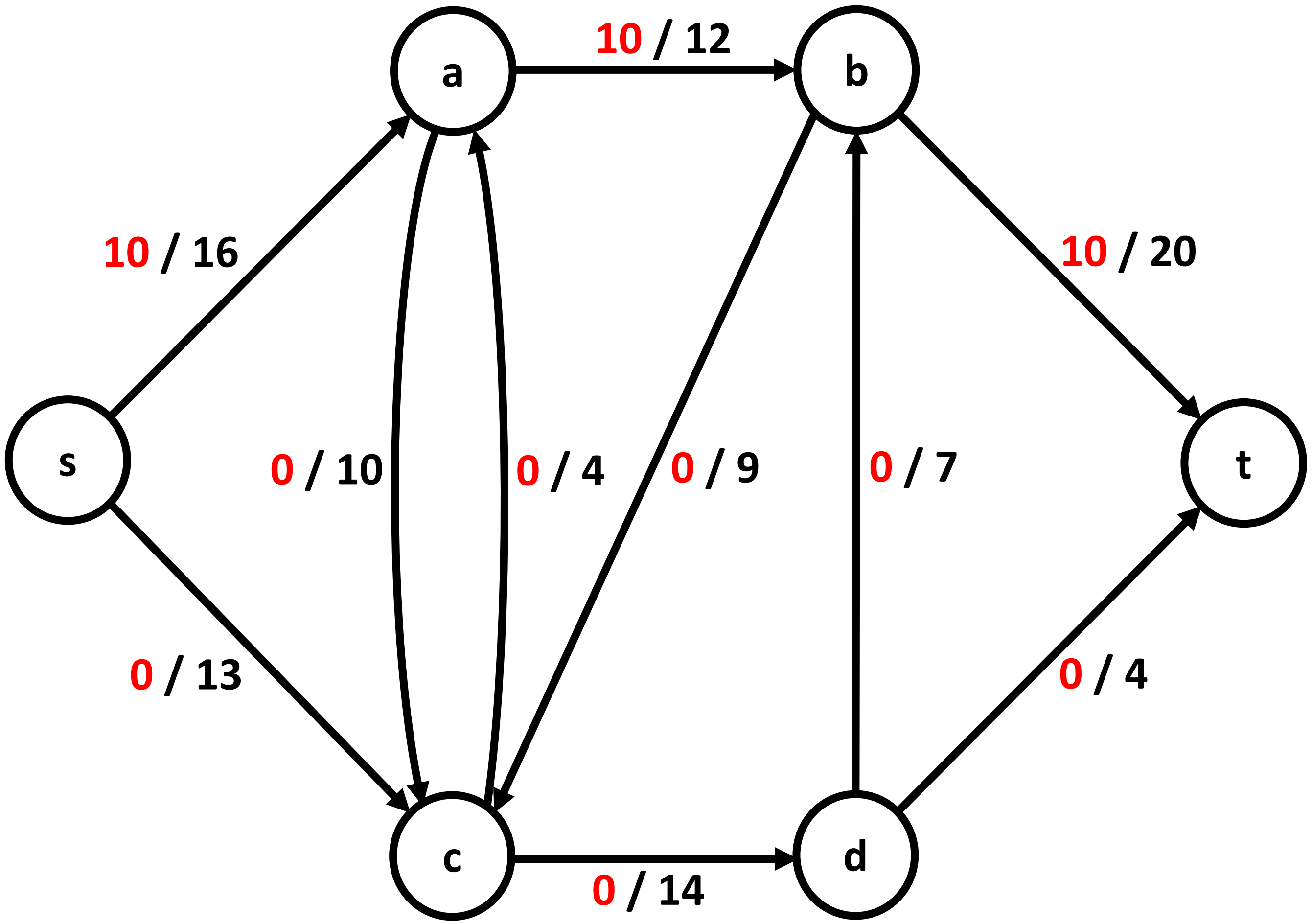
- Input: A directed graph, source vertex s, and sink vertex t. Each edge has a non-negative capacity.
- Assumption: No edge enters into s or leaves from t.



Maximum Flow Problem

- A st-flow (flow) is an assignment of values to the edges such that:
  - Capacity constraint: 0 ≤ edge’s flow ≤ edge’s capacity.
  - Flow constraint: inflow = outflow at every vertex (except s and t).
- The value of a flow is the inflow at t (or outflow from s).
- Maximum st-flow (max flow) problem: Find a flow of maximum value.
- Output: Find a flow of maximum value.

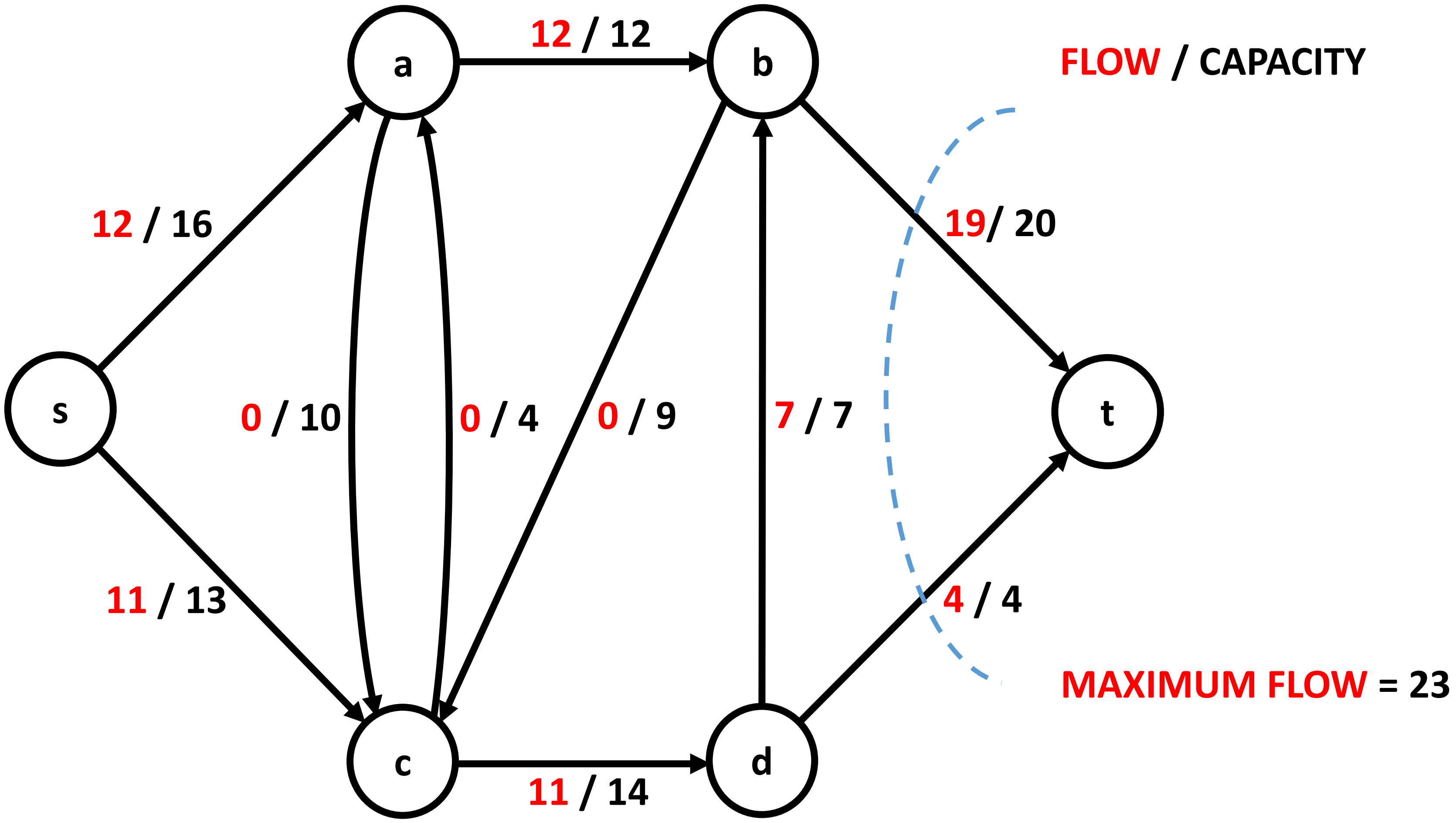
Flow vs Capacity



FLOW / CAPACITY

Maximum Flow

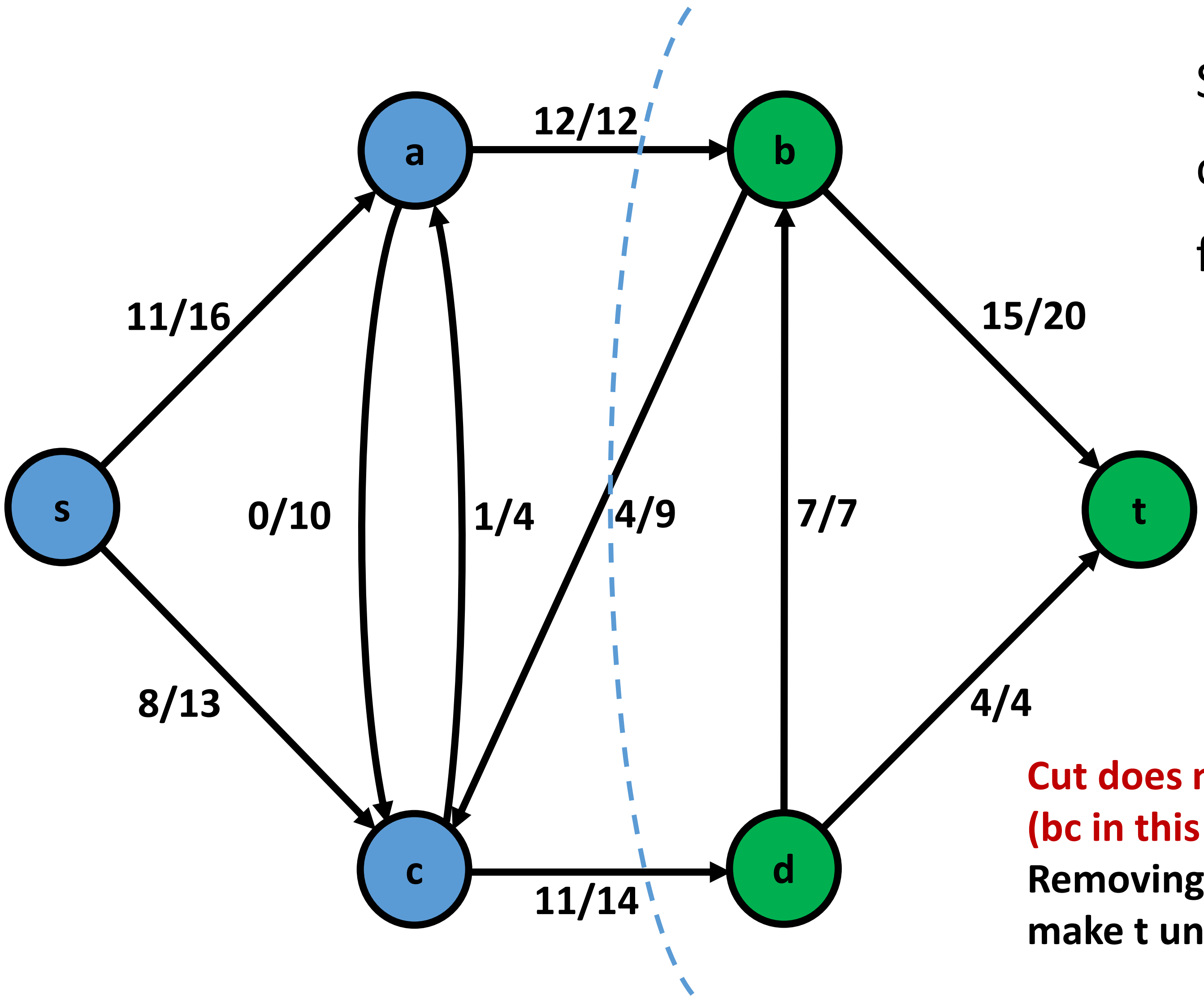
Inflow at b = 12 + 7 = 19
Outflow at b = 0 + 19 = 19



Minimum Cut Problem

- A st-cut is a partition of the vertices into two disjoint sets S and T with $s \in S$ and $t \in T$.
- Capacity of a st-cut $c(S, T)$ is the sum of the capacities of the edges from S to T.
- Cut does not count edges from T to S - Why? Removing forward edges is enough to make t unreachable from s.
- If f is a flow, then the net flow across the cut (S, T) is defined to be $f(S, T)$.
- Minimum st-cut (min cut) problem: Find a st-cut of minimum capacity.
- Output: Find a st-cut of minimum capacity.

Net Flow vs. Cut



S ={s, a, c} and T = {b, d, t}.

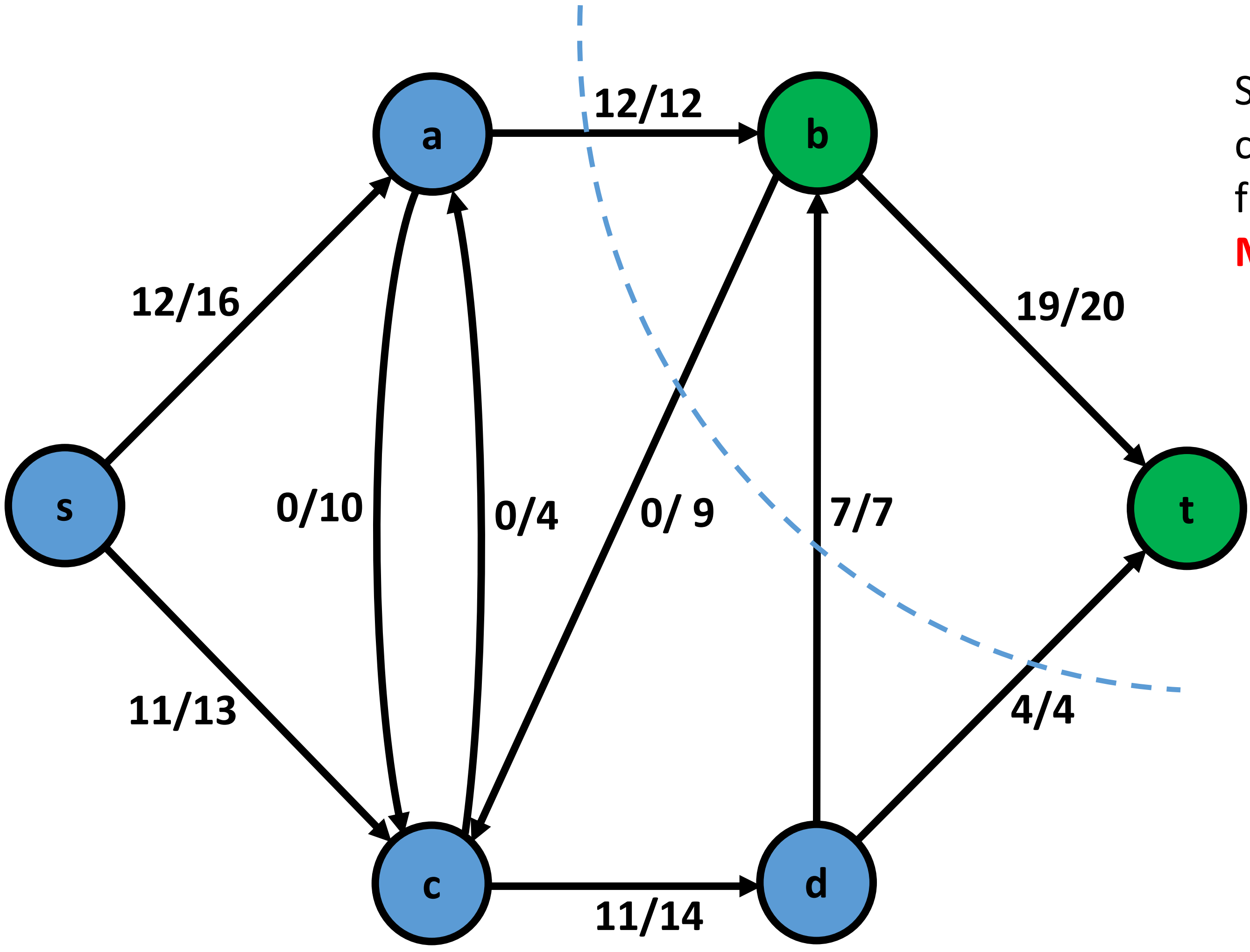
c(S, T) = 12 + 14 = 26.

f(S, T) = 12 – 4 + 11 = 19.

Cut does not count edges from T to S (bc in this example) Why?

Removing forward edges is enough to make t unreachable from s

Minimum Cut



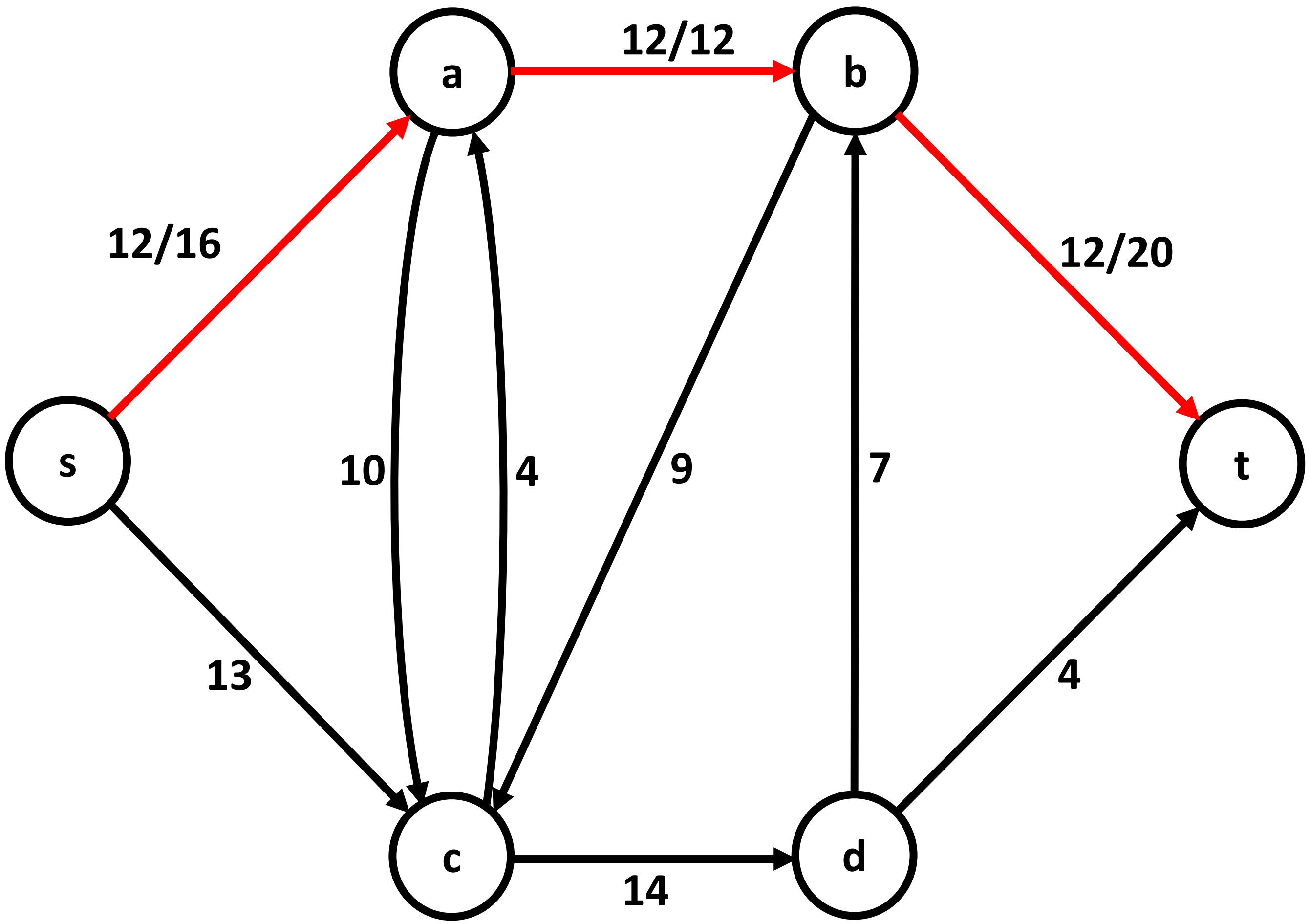
S ={s, a, c, d} and T = {b, t}.
c(S, T) = 12 + 7 + 4 = 23.
f(S, T) = 12 – 0 + 7 + 4 = 23.
MINIMUM CUT= 23

Few Definitions

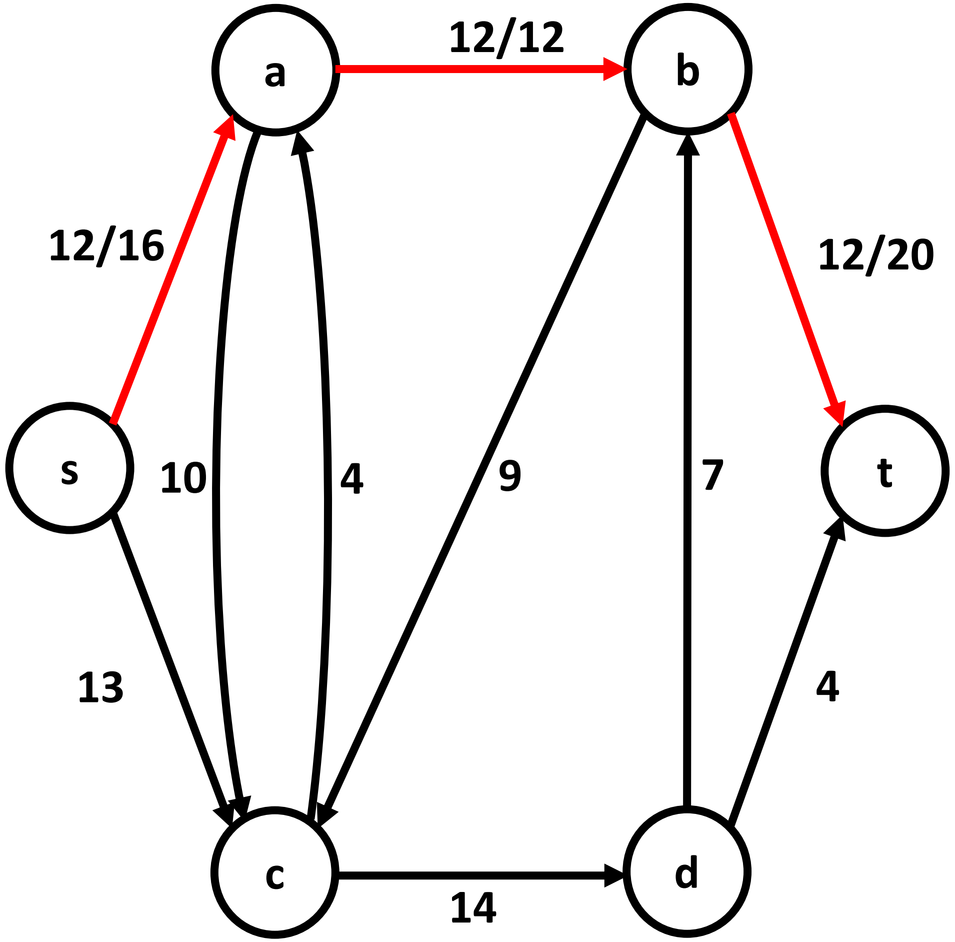
- Augmenting Path: An s,t path with positive capacity at each edge along the path
- Bottleneck Capacity: The maximum amount of fluid that can be flown in an augmenting path.
- Residual Network: Given a flow network $G=(V, E)$, the residual network is $G_R = (V, E_R)$ has edges (u, v) with weight

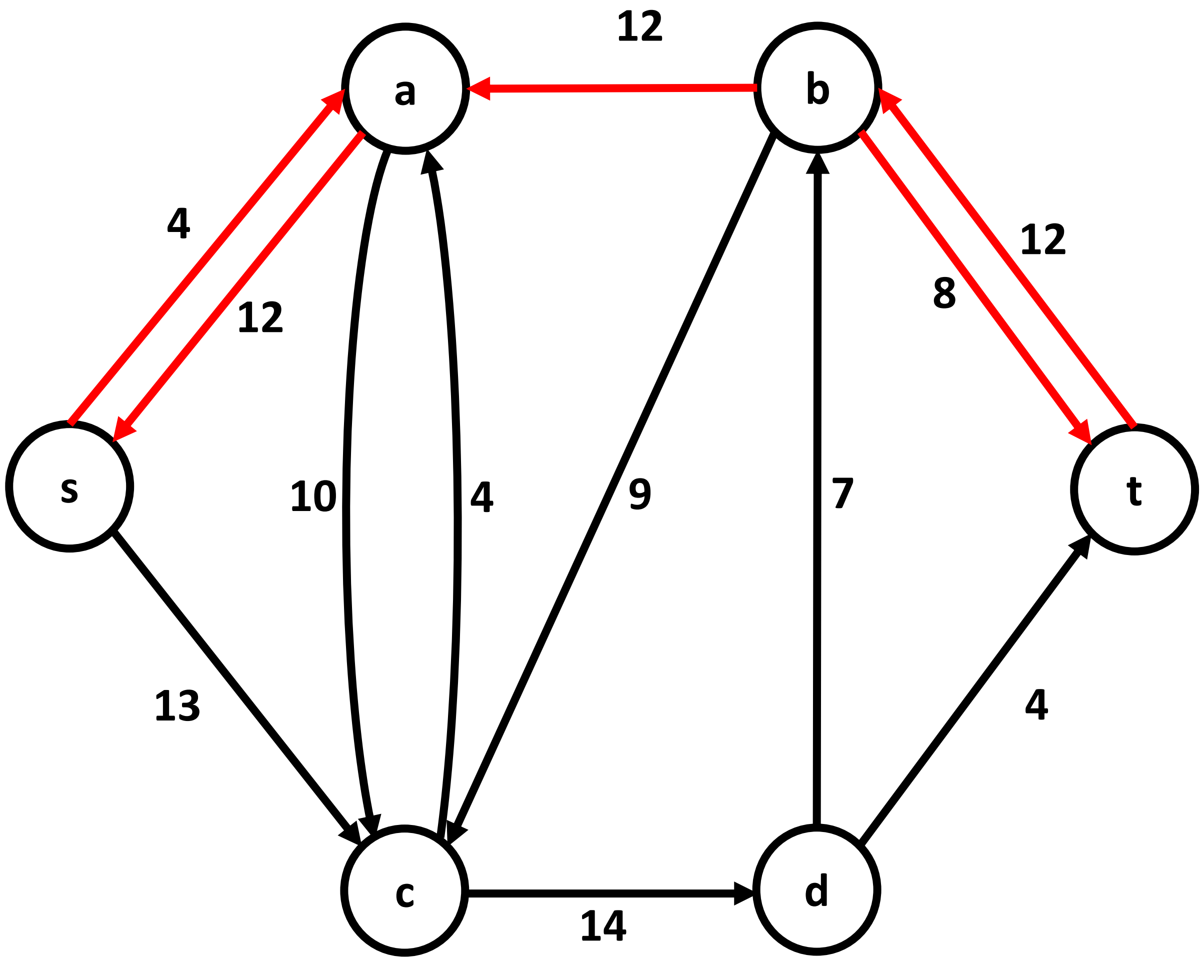
$$cap(u, v) - flow(u, v)$$

Bottleneck Capacity = Min(16, 12, 20) = 12



Residual Network





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 $E_f = \{(u, v) \in V \times V : c_f (u, v) > 0\}$.
- **Residual capacity** : $c_f (u, v) = c(u, v) – f(u, v)$
- Each edge of the residual network, or **residual edge**, can admit a flow $f > 0$.
- $|E_f | \leq 2 |E|$.

Proof: The edges in E_f are either edges in E or their reversals.

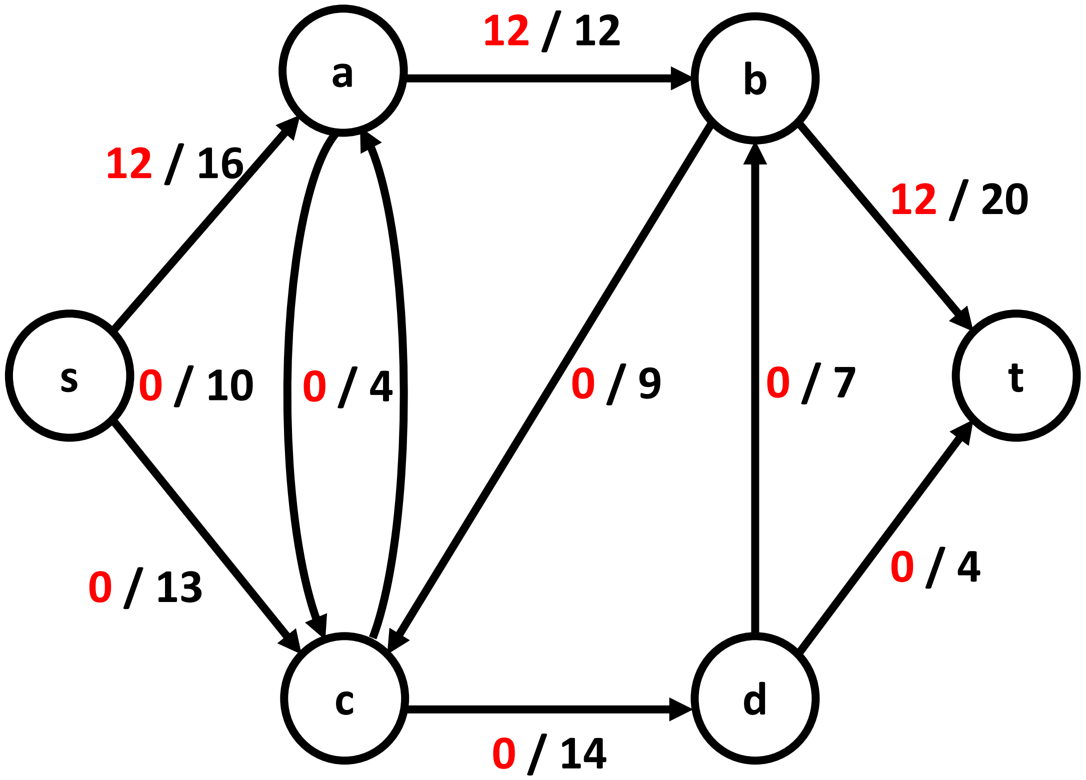
If $f(u, v) < c(u, v)$ for an edge $(u, v) \in E$, then $(u, v) \in E_f$ with $c_f(u, v) = c(u, v) - f (u, v) > 0$.

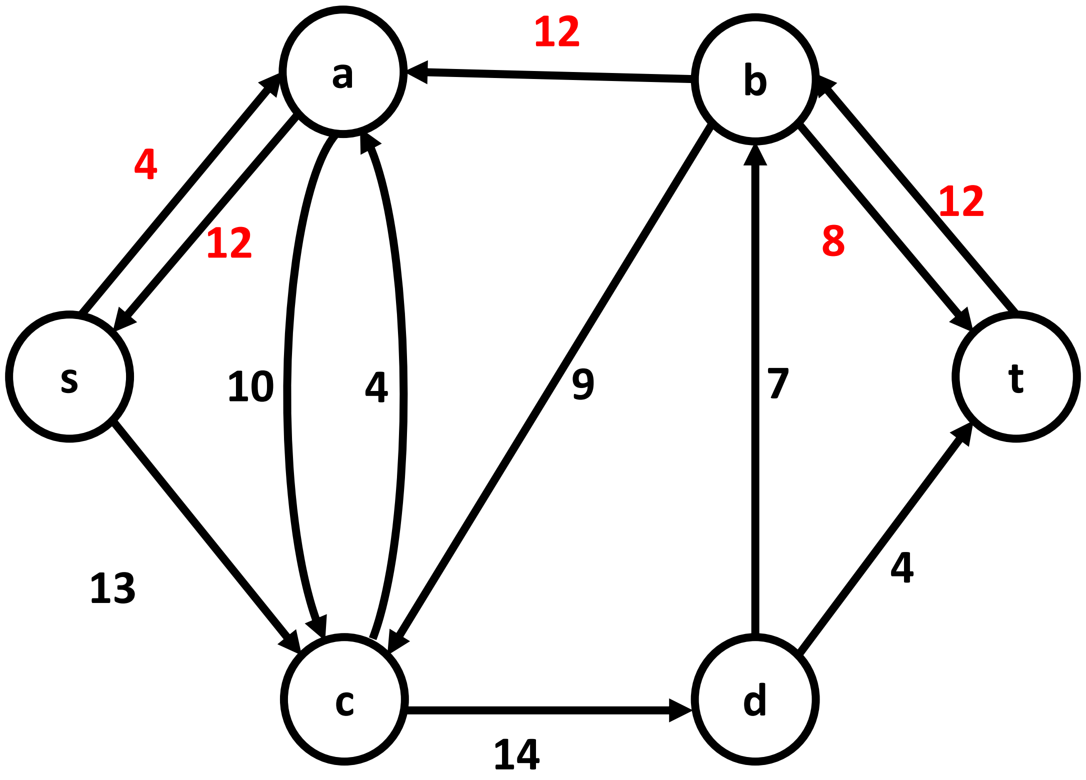
If $f(u, v) > 0$ for an edge $(u, v) \in E$, then $f(v, u) < 0$. Then, $(v, u) \in E_f$ with $c_f(v, u) = c(v, u) - f(v, u) > 0$.

If neither (u, v) nor (v, u) appears in E, then $c(u, v) = c(v, u) = 0$, $f (u, v) = f (v, u) = 0$. $c_f(u, v) = c_f(v, u) = 0$.

We conclude that an edge (u, v) can appear in a residual network only if at least one of (u, v) and (v, u) appears in the original network, and thus $| E_f | \leq 2 |E|$.

Flow Network vs Residual Network





Augmenting Path

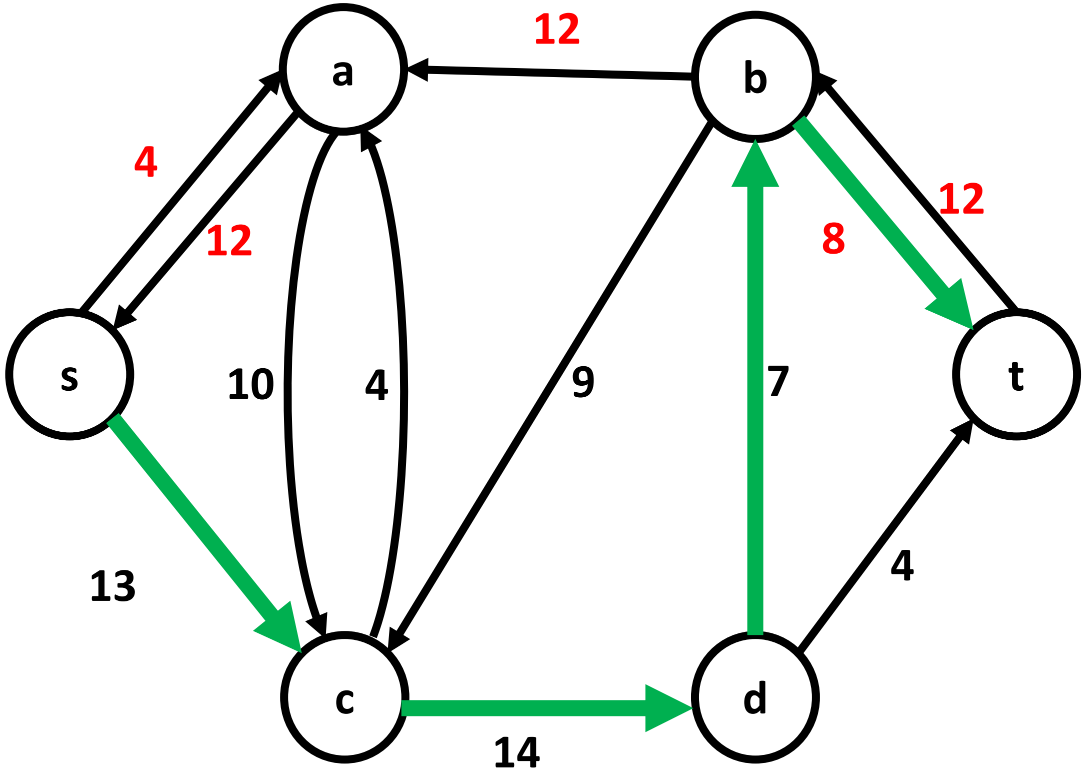
- Given a flow network $G = (V, E)$ and a flow f , an **augmenting path** p is a simple path from s to t in the residual network G_f.

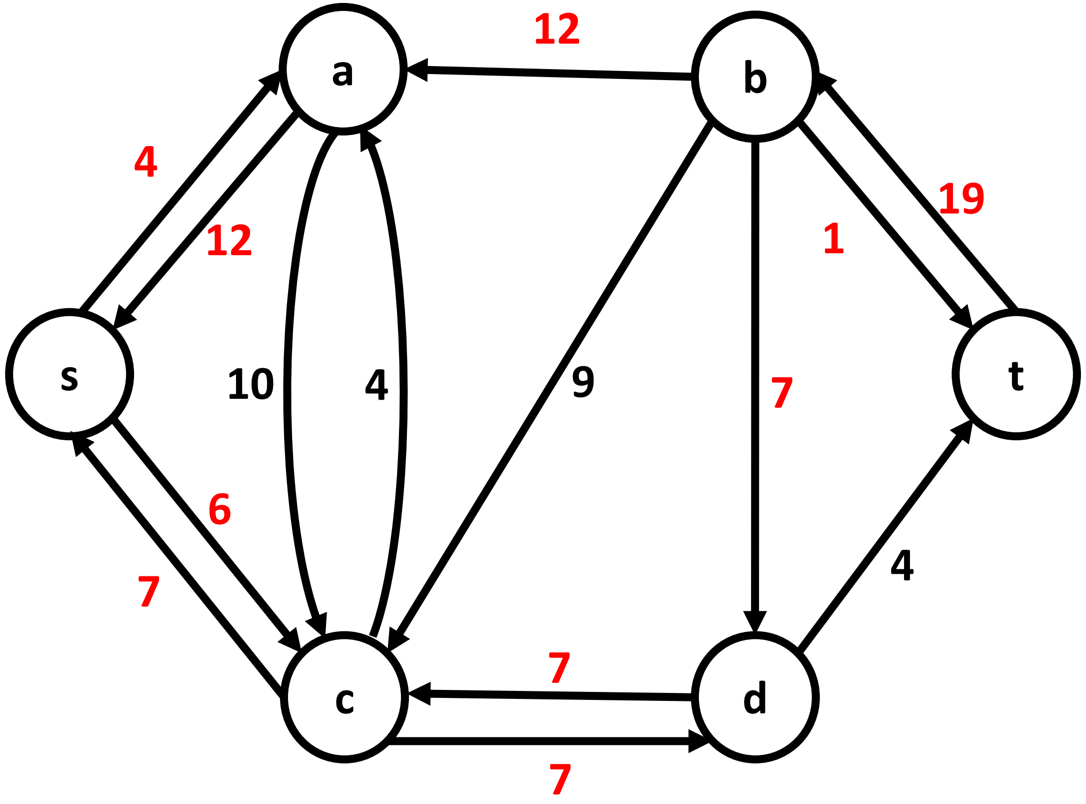
- By the definition of the residual network, each edge (u, v) on an augmenting path admits some additional positive flow from u to v without violating the capacity constraint on the edge.

- We call the maximum amount by which we can increase the flow on each edge in an augmenting path p, the **bottleneck capacity** of p, given by

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

Augmenting Path in Residual Network





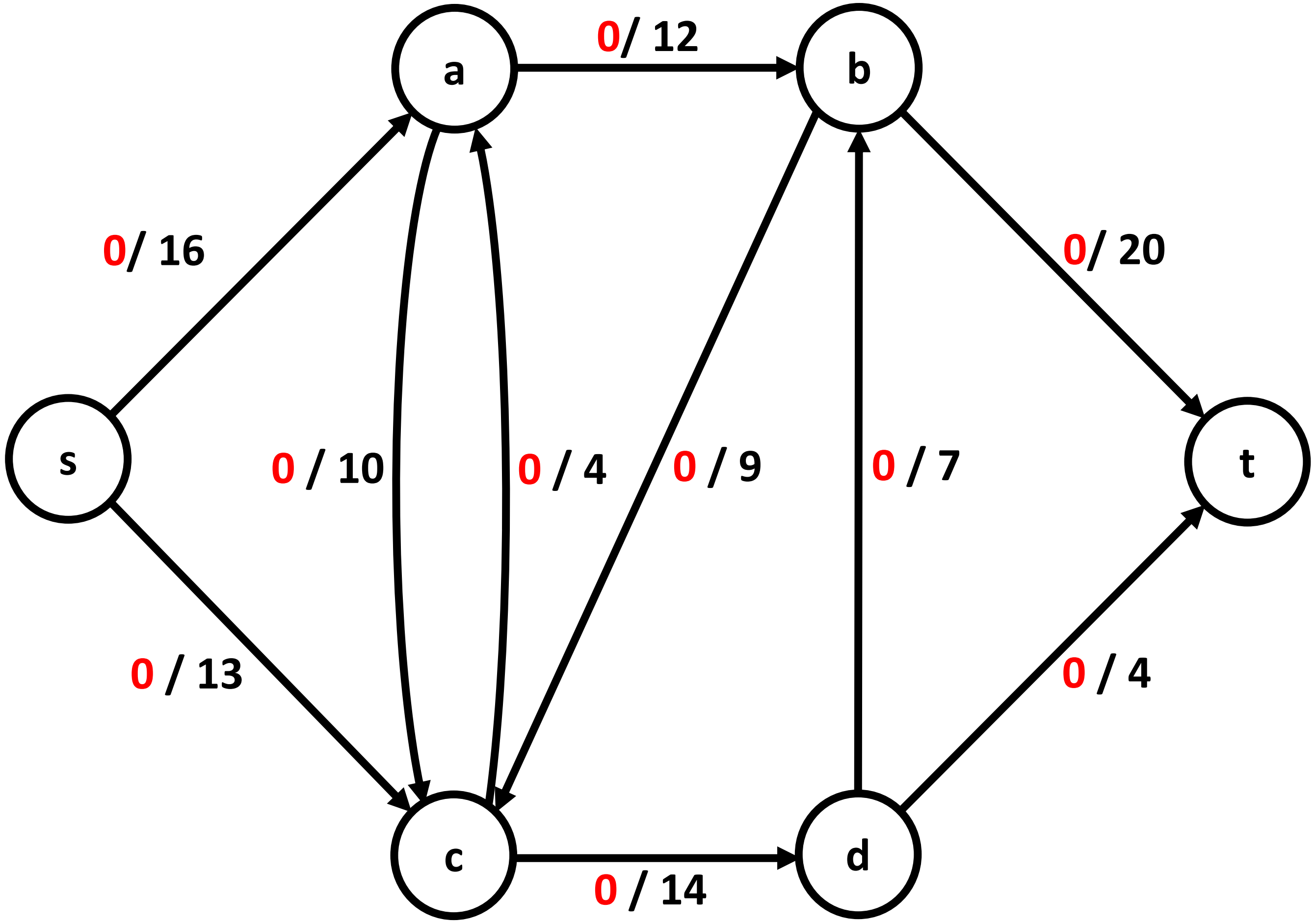
Bottleneck Capacity = min {13, 14, 7, 8} = 7

Ford-Fulkerson Algorithm

- Start with flow = 0.
- While there exists an augmenting path from s to t:
  - Find an augmenting path from s to t
  - Compute bottleneck capacity
  - Increase flow on that path by bottleneck capacity
  - Decrease capacity on that path by bottleneck capacity
- The variable flow gives the maximum flow.
- Run DFS to mark all reachable nodes from s. Call the set of vertices A.
- Cut edges of minimum size are from A to V – A.

Ford-Fulkerson Algorithm

Initialize Flow: Start with 0 flow.

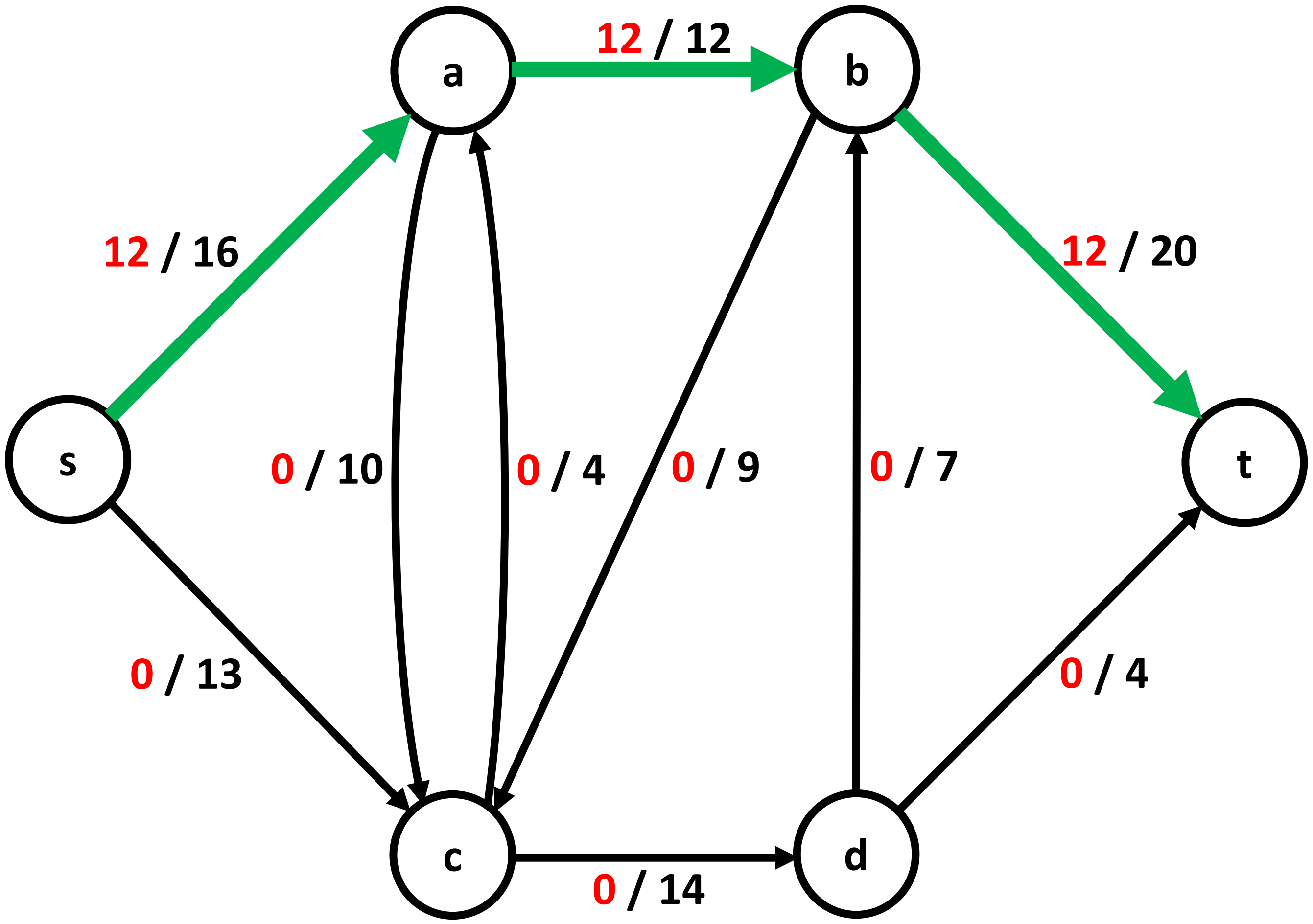


Flow = 0

Ford-Fulkerson Algorithm

Augmenting path. Find an undirected path from *s* to *t* such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).



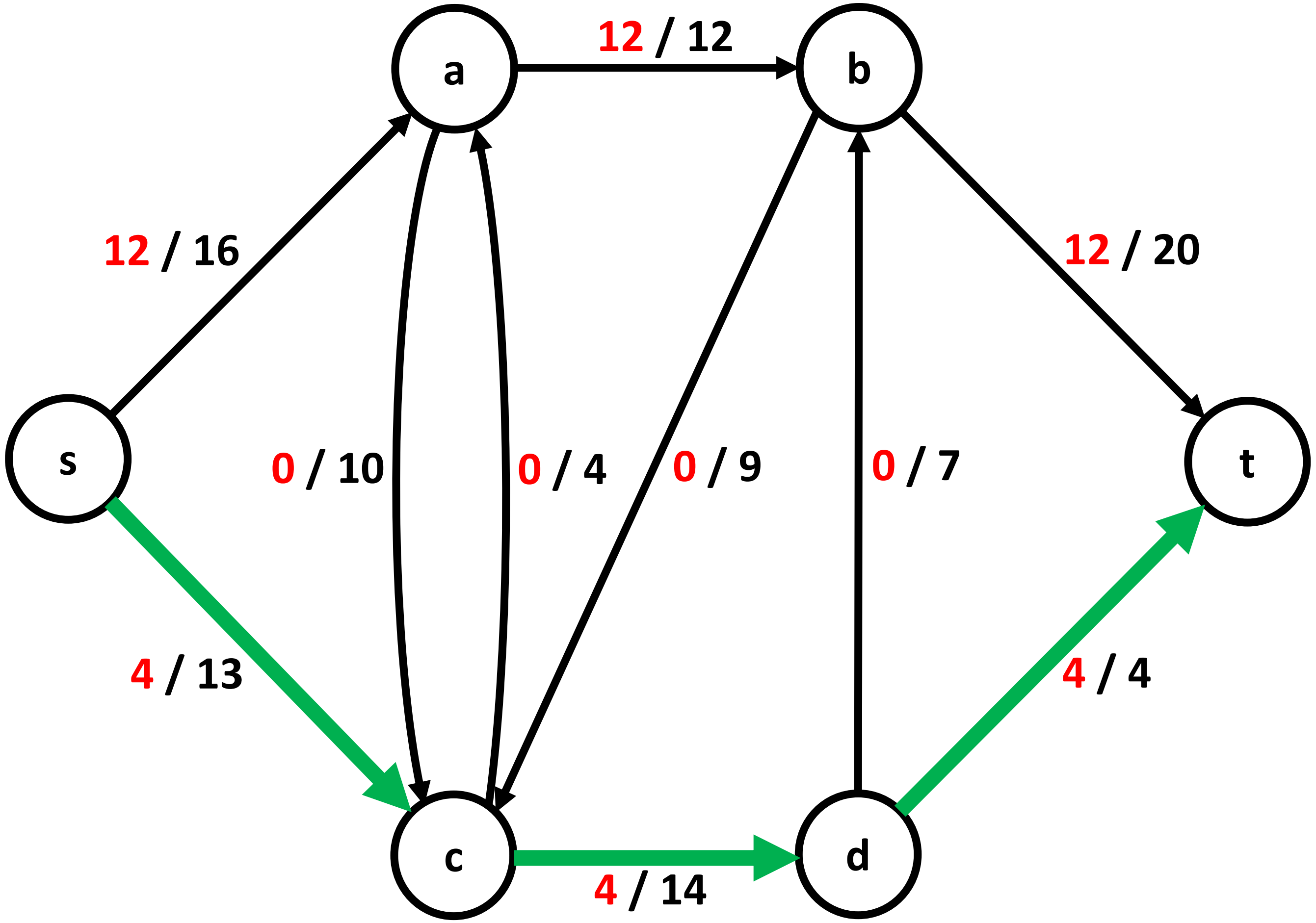
Bottleneck Capacity = 12

Flow = 0 + 12 = 12

Ford-Fulkerson Algorithm

Augmenting path. Find an undirected path from *s* to *t* such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).



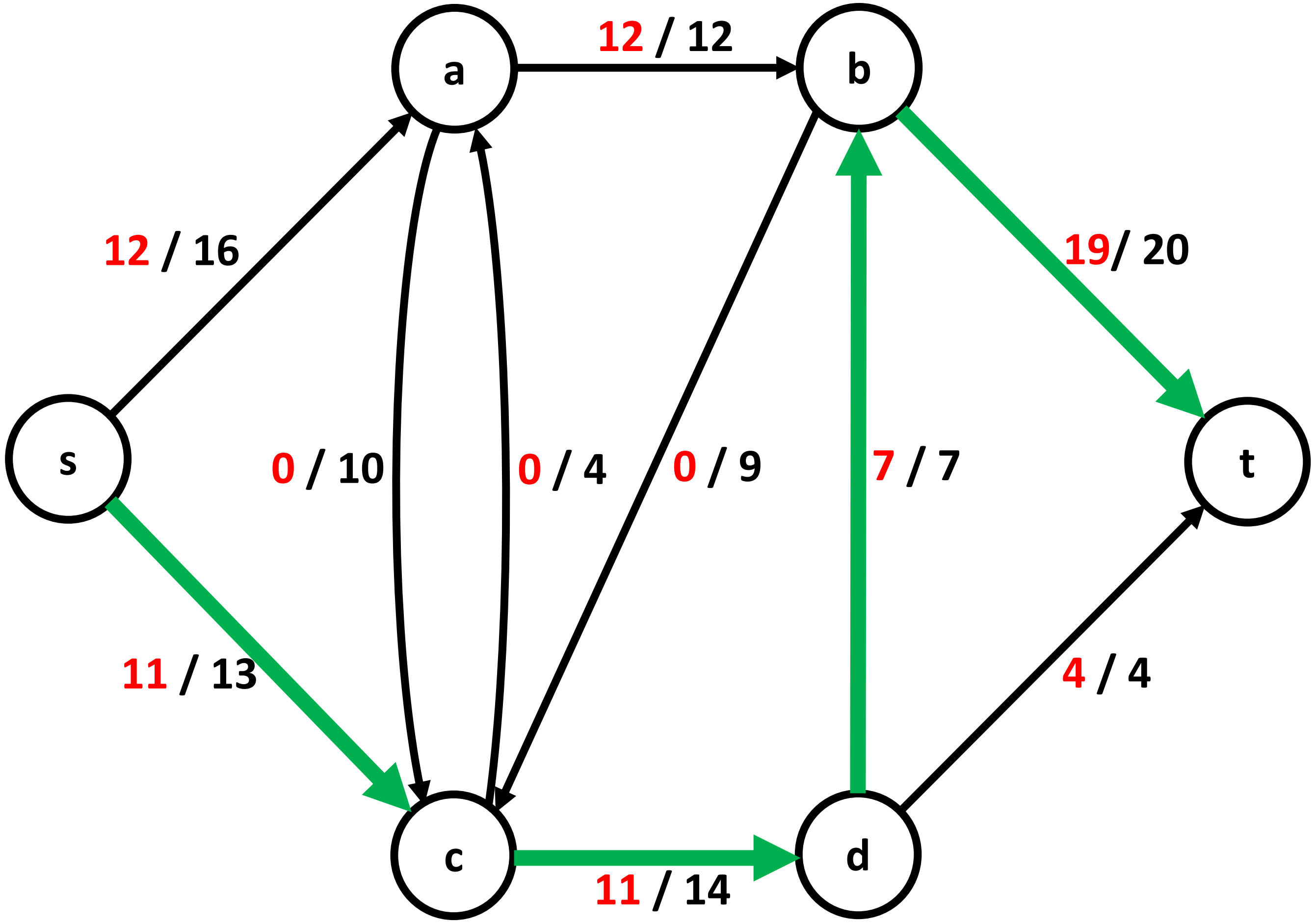
Bottleneck Capacity = 4

Flow = 0 + 12 + 4 = 16

Ford-Fulkerson Algorithm

Augmenting path. Find an undirected path from *s* to *t* such that:

- Can increase flow on forward edges (not full).
- Can decrease flow on backward edge (not empty).



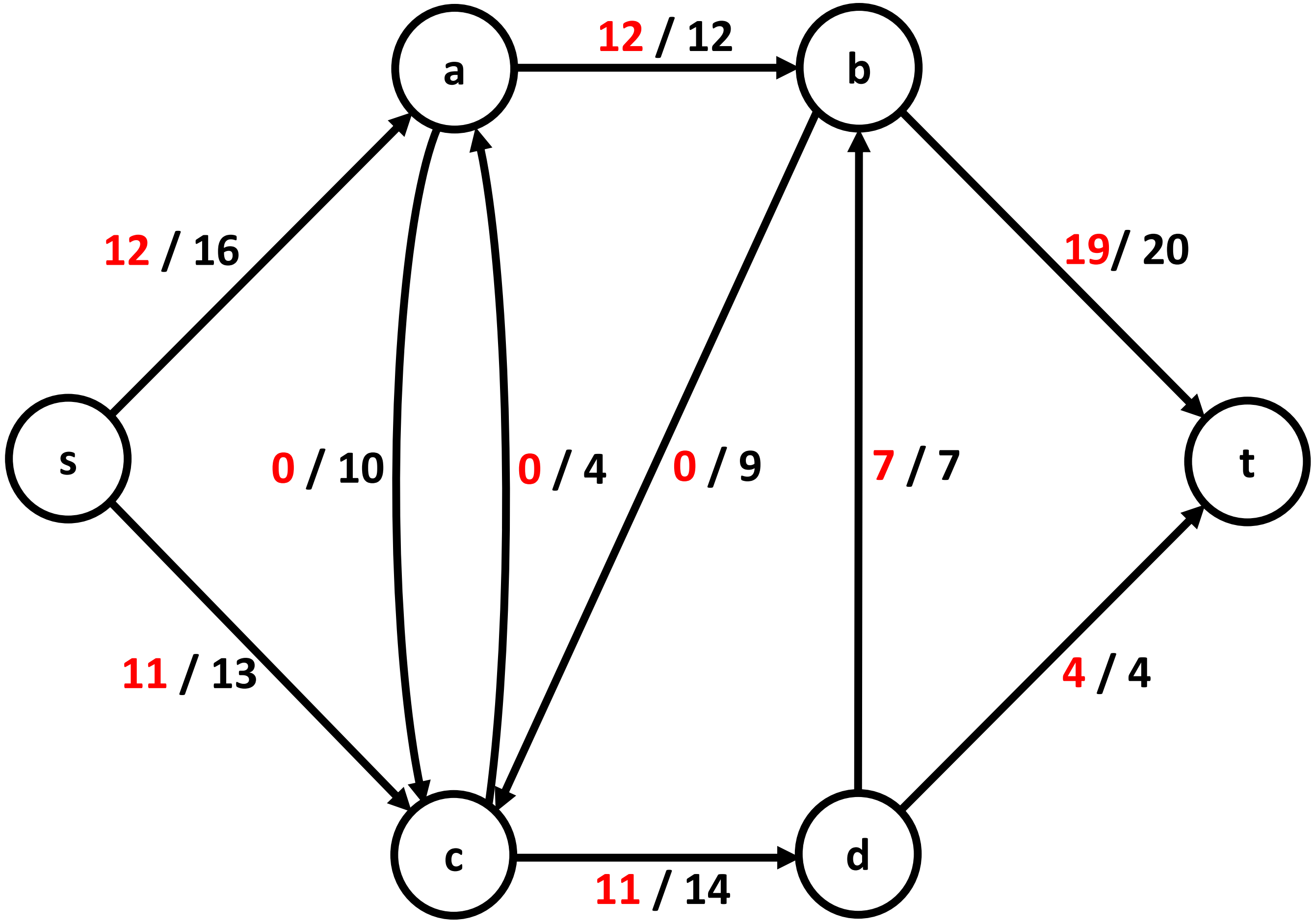
Bottleneck Capacity = 7

Flow = 0 + 12 + 4 + 7 = 23

Ford-Fulkerson Algorithm

Termination (No Augmenting path). All paths from *s* to t are blocked by either a

- Full forward edge.
- Empty backward edge.

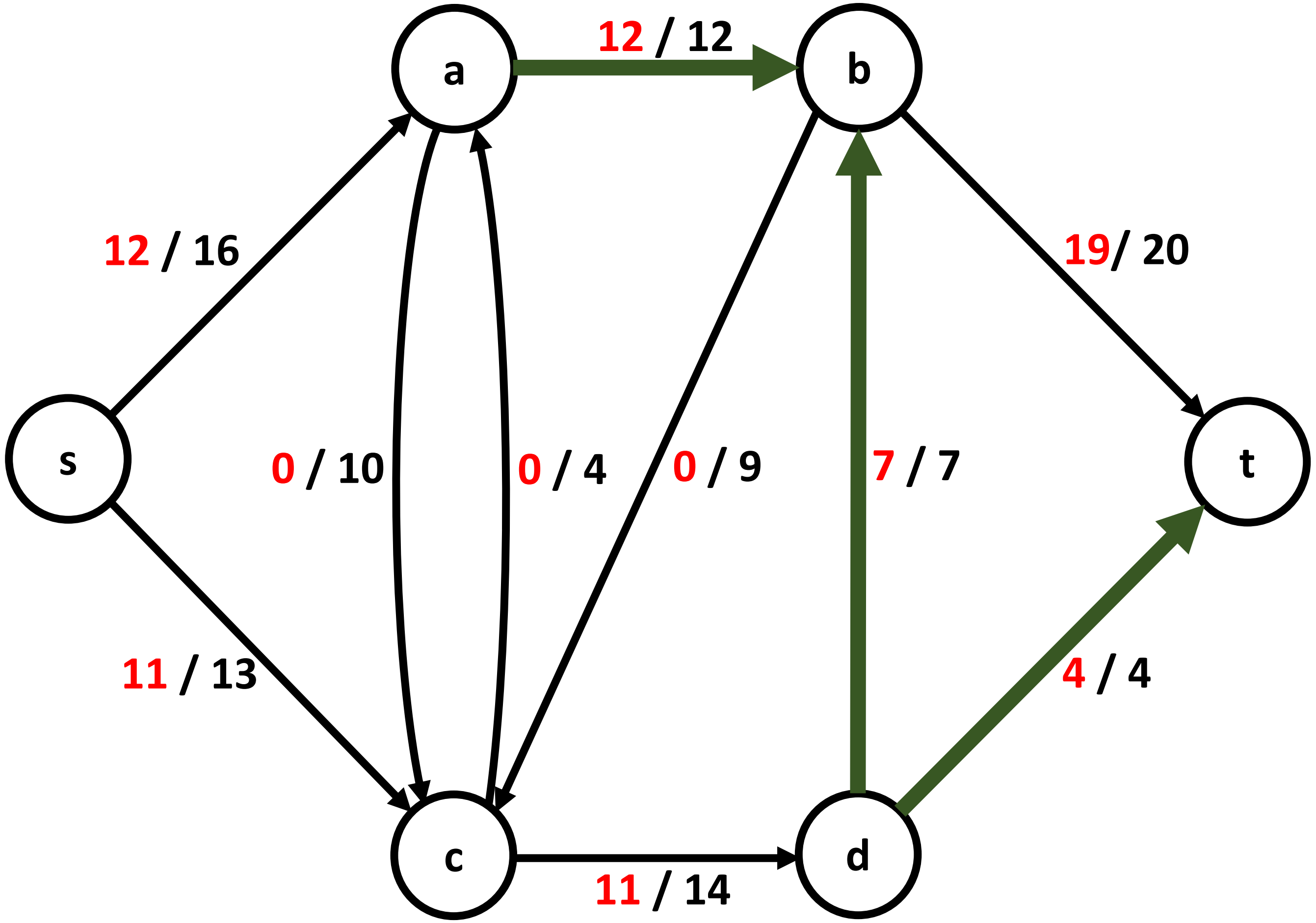


Bottleneck Capacity = 7

Flow = 0 + 12 + 4 + 7 = 23

Ford-Fulkerson Algorithm

Minimum Cut: DFS on residual graph



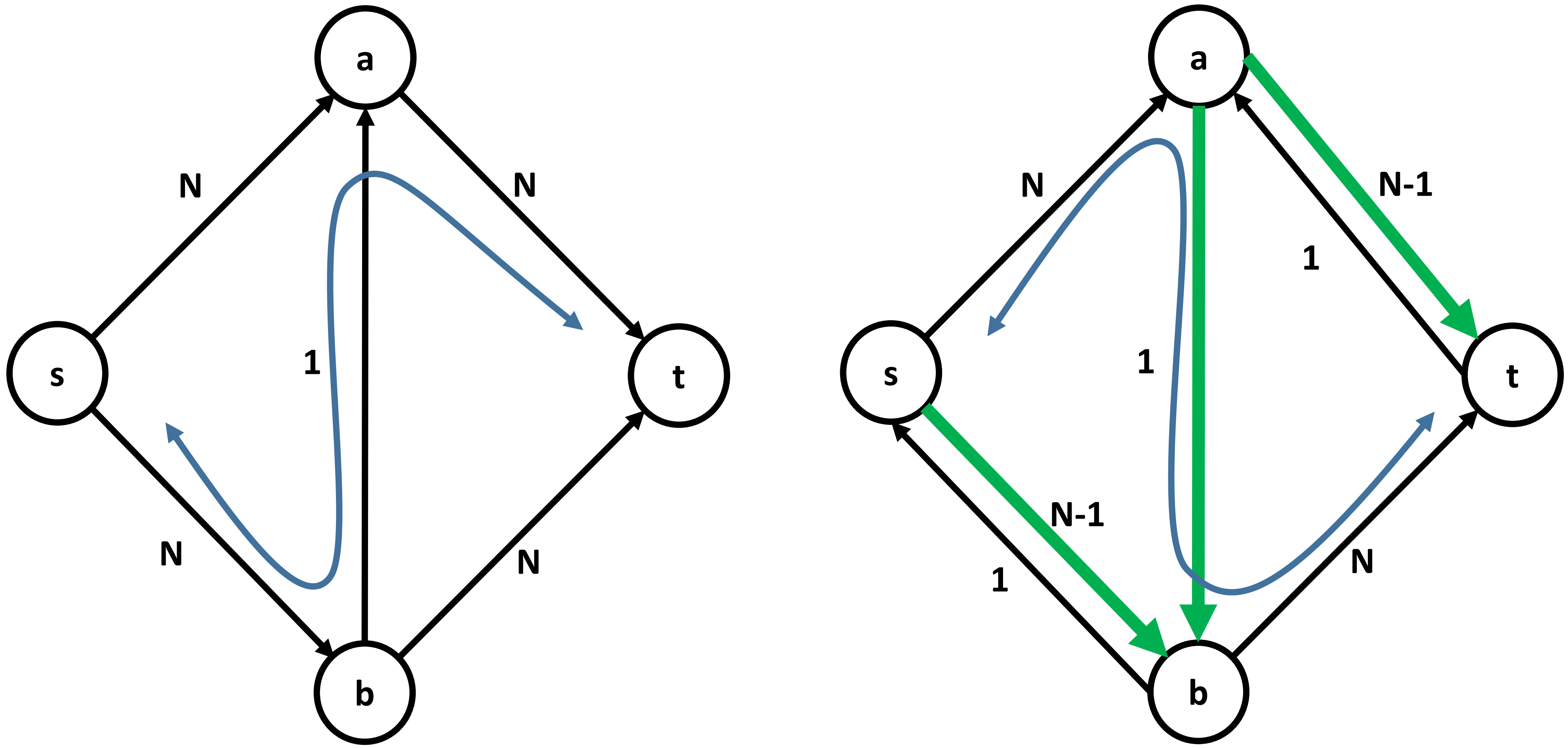
Cut = 12 + 7 + 4 = 23

Ford-Fulkerson Algorithm: Running Time

- Start with flow = 0. O(|V| + |E|)
- While there exists an augmenting path from s to t: O(f*)
  - Find an augmenting path using DFS from s to t O(|V| + |E|)
  - Compute bottleneck capacity
  - Increase flow on that path by bottleneck capacity
  - Decrease capacity on that path by bottleneck capacity
- The variable flow gives the maximum flow.
- Run DFS to mark all reachable nodes from s. Call the set of vertices A. O(|V| + |E|)
- Cut edges of minimum size are from A to V – A.

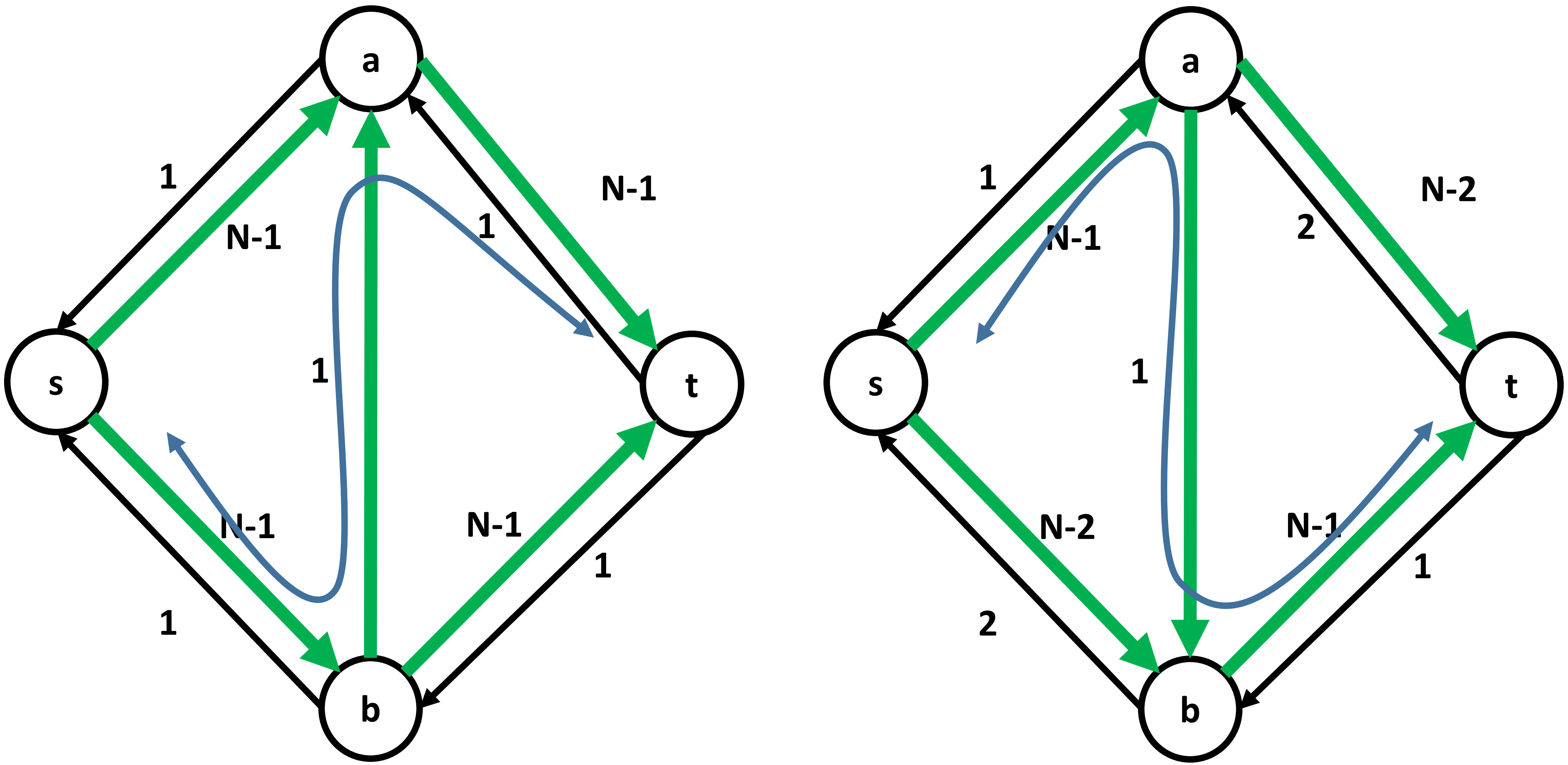
Ford-Fulkerson Algorithm: A Bad Case

Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maximum flow.



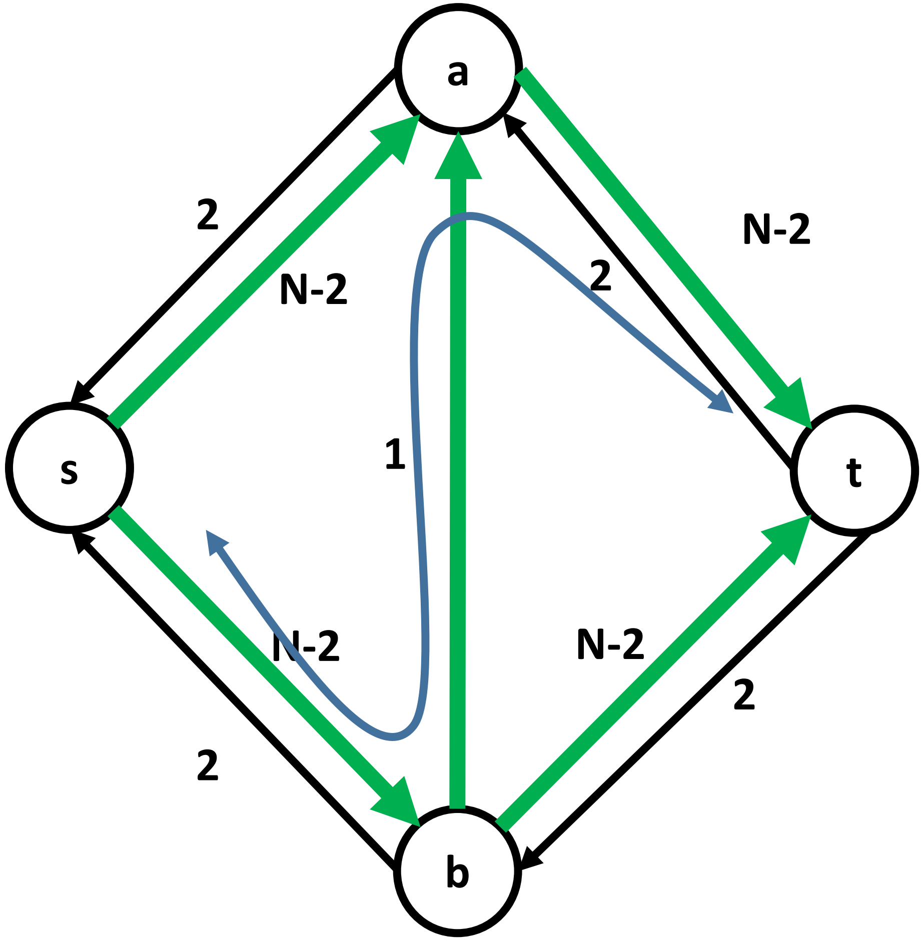
Ford-Fulkerson Algorithm: A Bad Case

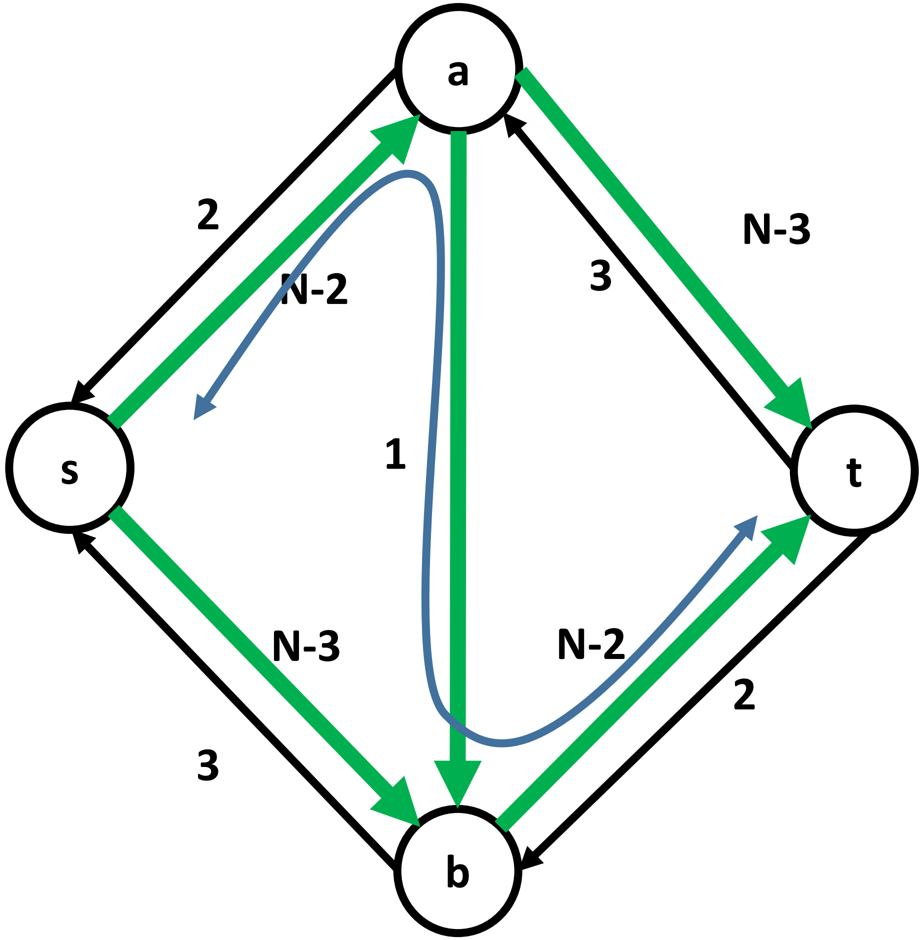
Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maximum flow.



Ford-Fulkerson Algorithm: A Bad Case

Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maximum flow.



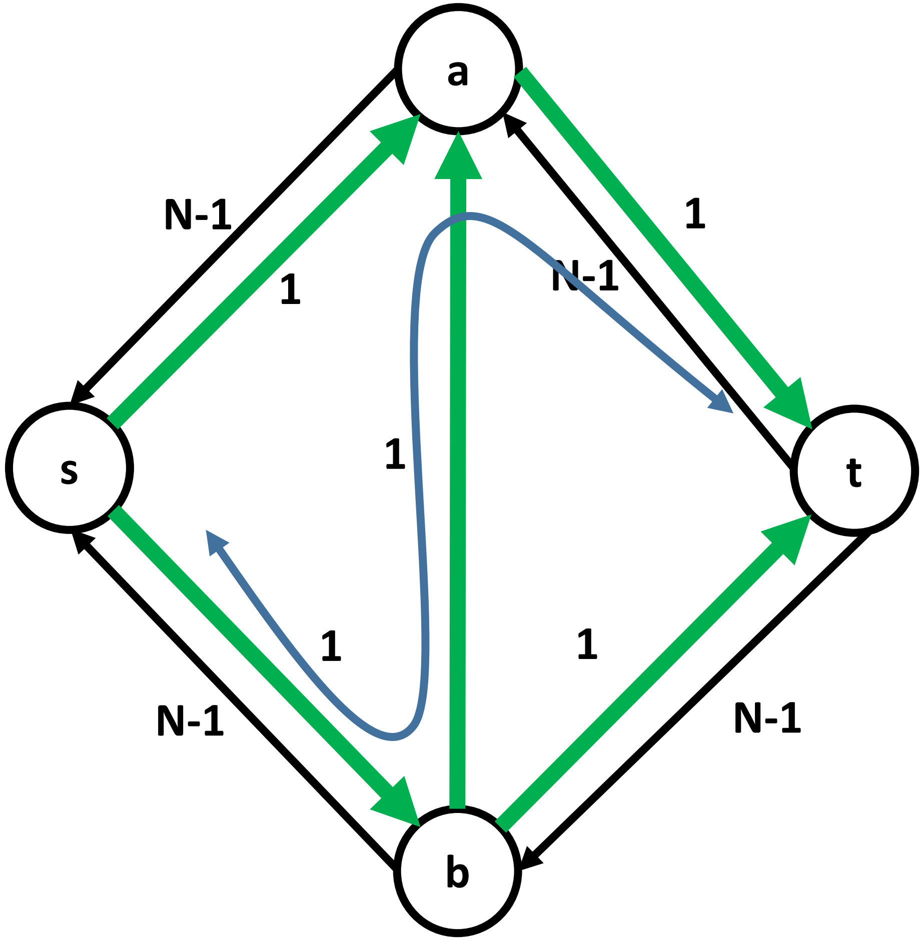


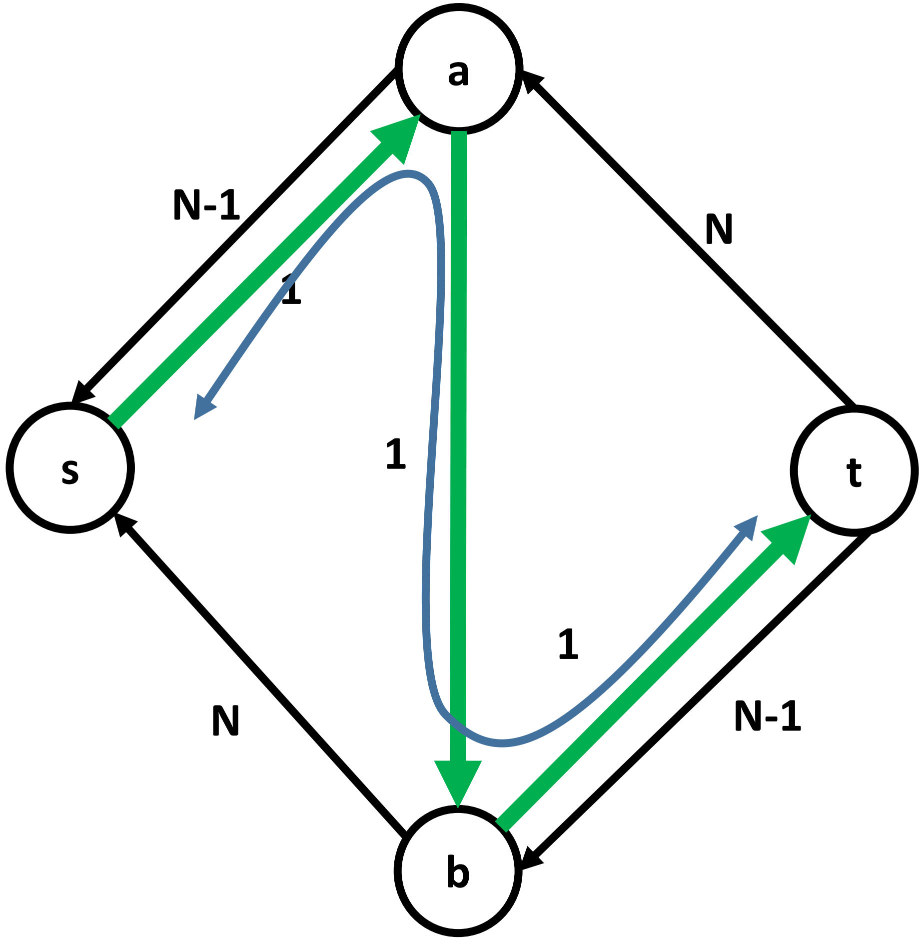
Ford-Fulkerson Algorithm: A Bad Case

Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maximum flow.

Ford-Fulkerson Algorithm: A Bad Case

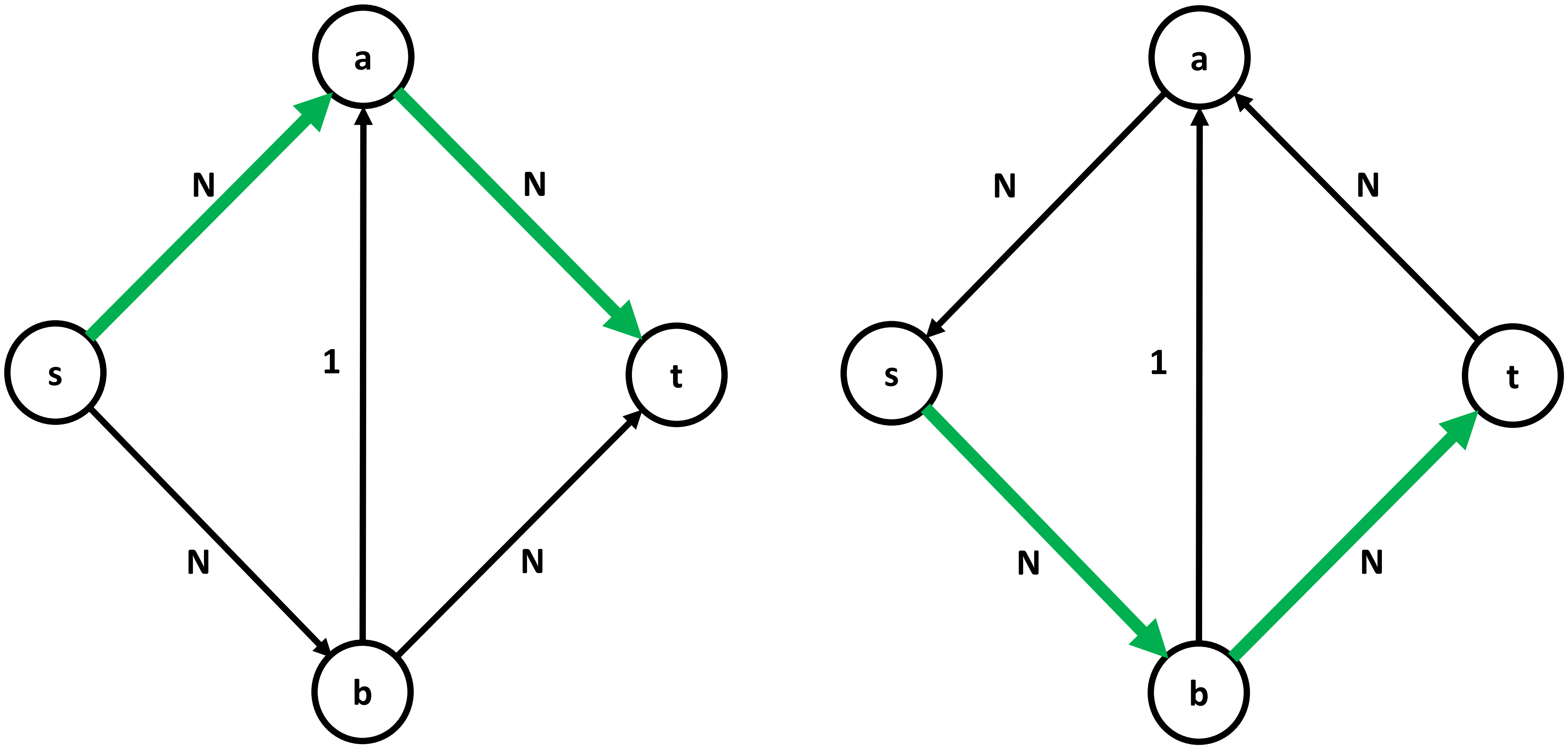
Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maximum flow.





Ford-Fulkerson Algorithm: Good News

This case can easily be avoided by using shortest path. Finding augmenting path by BFS from s to t.



Ford-Fulkerson Method

- It is a method proposed by Ford and Fulkerson, where they have suggested to find an augmenting path from s to t.
- It is not an algorithm since they do not mention about the exact way to find the augmenting path.
- DFS: $O(f^*(|V| + |E|))$
- BFS: $O(|V||E|(|V| + |E|))$ [Edmond]

- Implementation Note:

  Represent the residual network as an adjacency list.

Flow Network

- A flow network $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$. If (u, v) does not belong to E, $c(u, v) = 0$. Two special vertices are considered in a flow network: a source s and a sink t.
- A flow in G is a real-valued function $f : V \times V \rightarrow \mathbf{R}$ that satisfies the following three properties:
  - Capacity constraint**:** For all $u, v \in V$, we require $f(u, v) \leq c(u, v)$.
  - Skew symmetry**:** For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.
  - Flow conservation**:** For all $u \in V - \{s, t\}$, we require $\sum_{v \in V} f(u, v) = f(u, V) = 0$.
- The quantity $f(u, v)$, which can be positive, zero, or negative, is called the flow from vertex u to vertex v.
- The value of a flow f is defined as
  - $|f| = \sum_{v \in V} f(s, v) = f(s, V)$ (the total flow out of the source)

Flow Network

- The total positive flow entering a vertex v is defined by
  - $\sum_{u \in V} f(u, v)$ with $f(u, v) > 0$

- The total positive flow leaving a vertex u is defined by
  - $\sum_{v \in V} f(u, v)$ with $f(u, v) > 0$

- We define the total net flow at a vertex u to be the total positive flow leaving u minus the total positive flow entering u.
  - $\sum_{v \in V} f(u, v)$ with $f(u, v) > 0 - \sum_{v \in V} f(v, u)$ with $f(v, u) > 0$

- Notation: $f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$

Properties of Flow Network

- $f(u, u) = 0$ for all $u \in V$

Proof: **Skew symmetry:** For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.

Put $v = u$.

Then $f(u, u) = -f(u, u)$.

This implies that $f(u, u) = 0$.

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

Proof: **Skew symmetry:** For all $u, v \in V$, we require $f(v, u) = -f(u, v)$.

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

$f(V, u) = -f(u, V)$.

Flow conservation: For all $u \in V - \{s, t\}$, we require $f(u, V) = 0$.

This implies that $f(V, u) = 0$.

Properties of Flow Network

- Let $G = (V, E)$ be a flow network, and let f be a flow in G. Then
    1. For all $X \subseteq V$, we have $f(X, X) = 0$

Proof: **Skew symmetry:** For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.

$\sum_{u \in X} \sum_{v \in X} f(u, v) = -\sum_{v \in X} \sum_{u \in X} f(v, u)$. [Since $X \subseteq V$]

Then $f(X, X) = -f(X, X)$.

This implies that $f(X, X) = 0$.

- Let $G = (V, E)$ be a flow network, and let f be a flow in G. Then
    2. For all $X, Y \subseteq V$, we have $f(X, Y) = -f(Y, X)$.

Proof: **Skew symmetry:** For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.

$\sum_{u \in X} \sum_{v \in Y} f(u, v) = -\sum_{v \in Y} \sum_{u \in X} f(v, u)$. [Since $X, Y \subseteq V$]

$\sum_{u \in X} f(u, Y) = -\sum_{v \in Y} f(v, X)$.

This implies that $f(X, Y) = -f(Y, X)$.

Properties of Flow Network

- Let $G = (V, E)$ be a flow network, and let f be a flow in G. Then
  3. For all $X, Y, Z \subseteq V$ with $X \cap Y = \emptyset$, we have
     (i) $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ and
     (ii) $f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$.

Proof: (i) $f(X \cup Y, Z)$

$= \sum_{u \in X \cup Y} f(u, Z)$

$= \sum_{u \in X} f(u, Z) + \sum_{u \in Y} f(u, Z)$ [Since $X \cap Y = \emptyset$]

This implies that $f(X, Z) + f(Y, Z)$.

(ii) $f(Z, X \cup Y)$

$= \sum_{u \in X \cup Y} f(Z, u)$

$= \sum_{u \in X} f(Z, u) + \sum_{u \in Y} f(Z, u)$ [Since $X \cap Y = \emptyset$]

This implies that $f(Z, X) + f(Z, Y)$.

Properties of Flow Network

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

Proof: $| f | = f(s, V) =$

$= f(V, V) – f(V – s, V)$ [Part 3]

$= – f(V – s, V)$ [Part 1]

$= f(V, V – s)$ [Part 2]

$= f(V, V – \{s, t\}) + f(V, t)$ [By flow conservation, $f(V, V – \{s, t\}) = 0$ how?]

$= f(V, t)$

[By flow conservation, for all $u \in V – \{s, t\}$, we have $f(u, V) = 0$

This implies that $\sum_{u \in V - \{s, t\}} = f(u, V) = 0$

$f(V – \{s, t\}, V) = 0$

Again, $f(V, V – \{s, t\}) = – f(V – \{s, t\}, V) = 0$]

Flow Cut Lemma

- Let f be a flow in a flow network G with source s and sink t, and let (S, T) be a cut of G. Then the net flow across (S, T) is $f(S, T) = |f|$.

Proof: $f(S, T) = f(S, V - S)$

$= f(S, V) - f(S, S)$ [Part 3]

$= f(S, V)$

$= f(s, V) + f(S - s, V)$ [Part 3]

$= f(s, V)$ [By flow conservation $f(S - s, V) = 0$]

$= |f|$

Weak Duality Property of Flow Network

- The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Proof: Let (S, T) be any cut of G and let f be any flow.

$| f | = f(S, T)$ [by flow cut lemma]

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

$\leq \sum_{u \in S} \sum_{v \in T} c(u, v)$ [by capacity constraint]

$= c(S, T)$

Max-flow min-cut theorem

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

Proof: (1) $\Rightarrow$ (2): By contradiction. f is a maximum flow in G but G_f has an augmenting path p.

Then, some more flow f_p can be shipped through p, resulting to increase in flow from f to $f + f_p$.

(2) $\Rightarrow$ (3): Suppose that G_f has no augmenting path, that is, that G_f contains no path from s to t.

Define $S = \{v \in V : \text{there exists a path from } s \text{ to } v \text{ in } G_f\}$ and $T = V - S$.

The partition (S, T) is a cut where $s \in S$ and t does not belong to S, otherwise there is a path from s to t in G_f. For each pair of vertices $u \in S$ and $v \in T$, we have $f(u, v) = c(u, v)$, since otherwise $(u, v) \in E_f$, which would place v in set S. Thus, $f(S, T) = c(S, T)$. By flow cut lemma, $|f| = f(S, T)$. Therefore, $|f| = c(S, T)$.

(3) $\Rightarrow$ (1): By weak duality lemma, $|f| \le c(S, T)$ for all cuts (S, T). The condition $|f| = c(S, T)$ thus implies that f is a maximum flow.