國立臺灣大學
National Taiwan University

CHAPTER 1
NETWORK FLOW

Iris Hui-Ru Jiang
Fall 2017

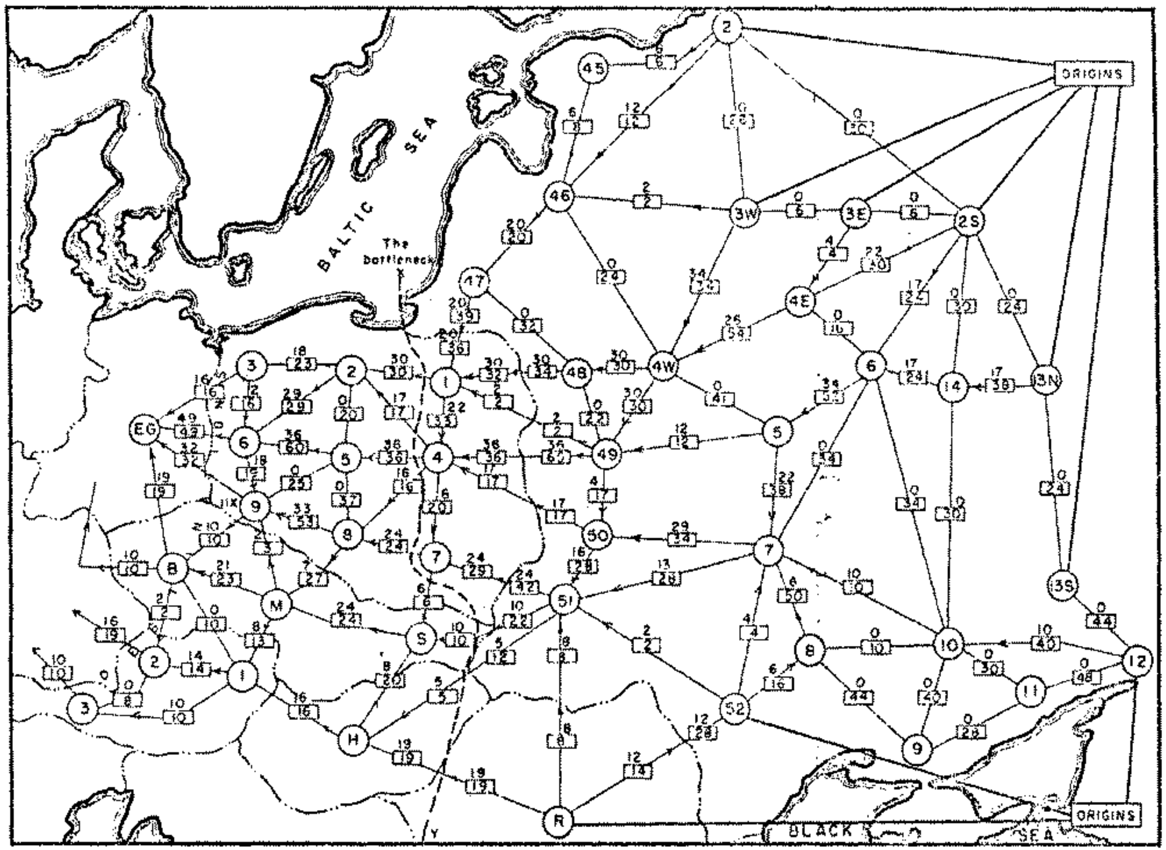
Department of Electrical Engineering
National Taiwan University

Outline

- Content:
  - Network flow
  - Bipartite matching: a special case of network flow
- Reading:
  - Chapter 7

Flow Network (1/2)

- Abstraction for material flowing through the edges.
  - Water pipes: water
  - Transportation network: traffic
  - Computer network: packets
  - Circuit network (wires): current



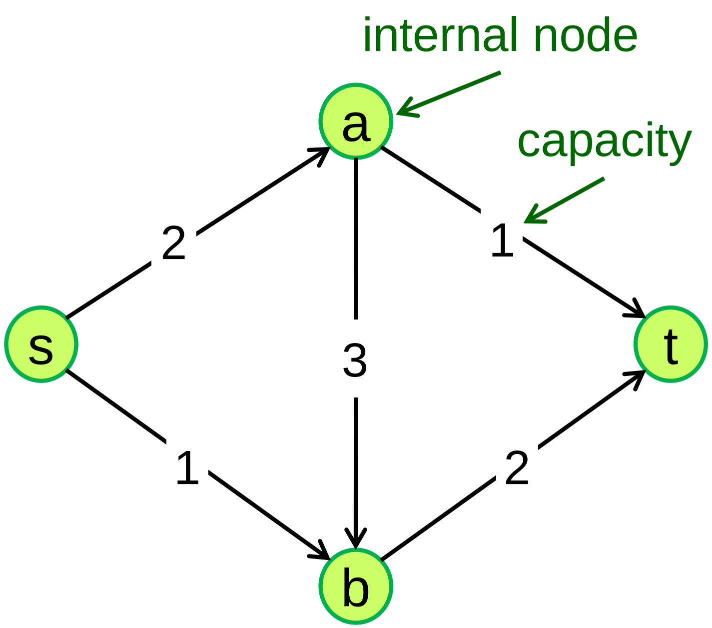
Harris & Ross, 1955

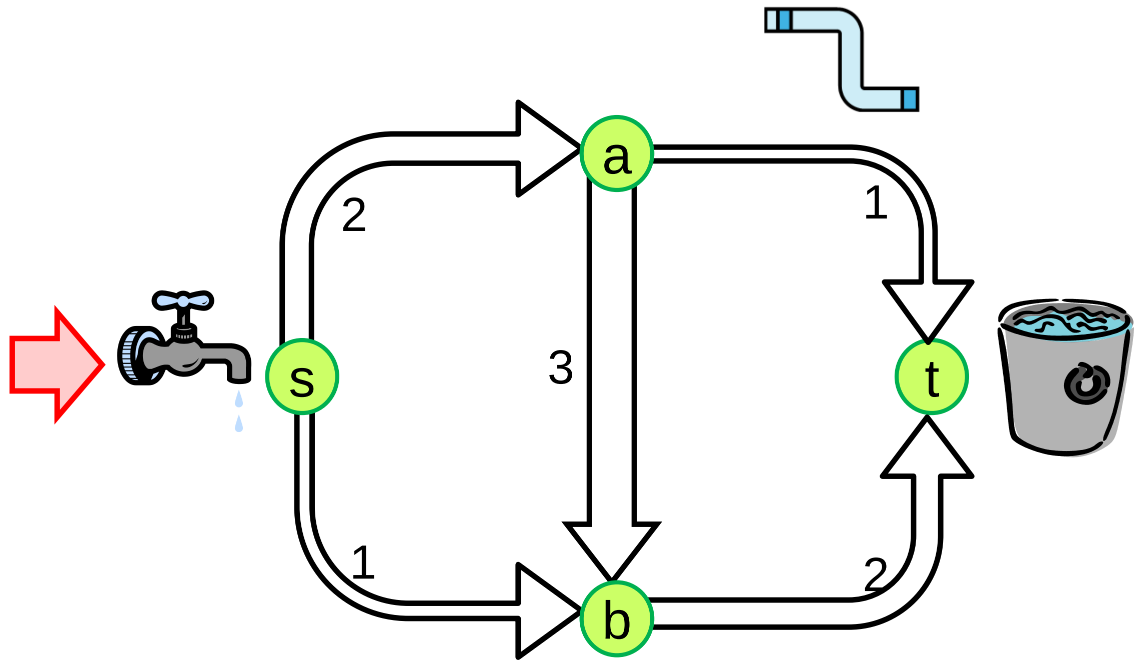
The Soviet and Eastern European railways network: 44 nodes and 105 undirected edges

Maximum flow:
163,000 from Russia to Eastern Europe
=
Bottleneck (cut of capacity):
163,000

Flow Network (2/2)

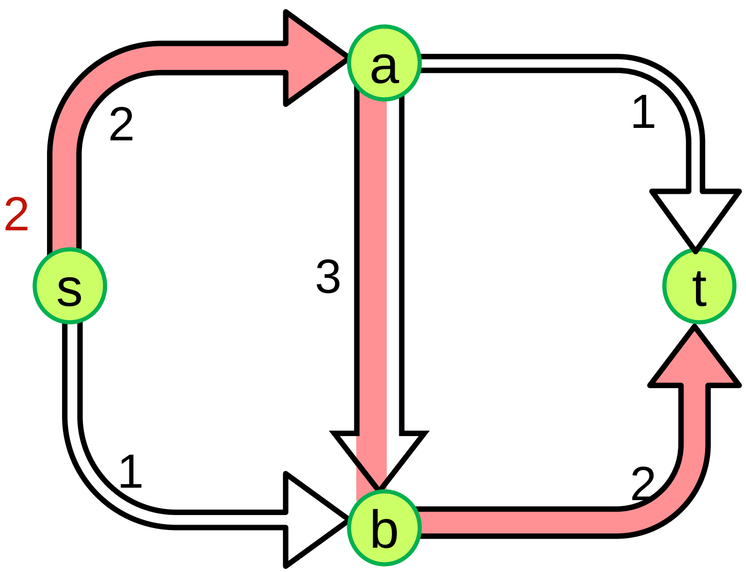
- A flow network $G = (V, E)$ is a directed graph
  - $c_e \geq 0$: capacity of edge e
  - Source $s \in V$: generates the flow
  - Sink $t \in V$: absorbs the flow
    - Internal node: $u \in V \setminus \{s, t\}$

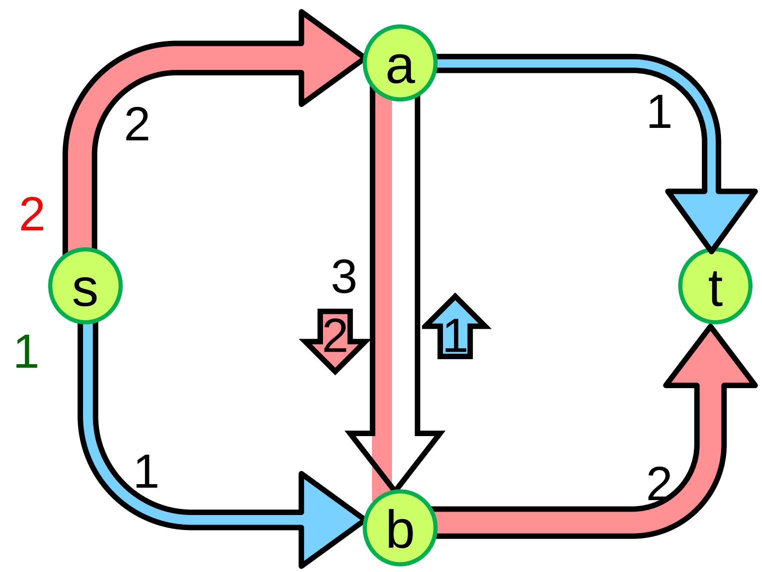




Pushing Flow

- **Greedy**
  - Start with zero flow
  - Push a flow of value 2
  - ⇒ Flow = 2
  - Q: Can we push more?
- **General**
  - Start with zero flow
  - Push a flow of value 2
  - Push a flow of value 1
  - A: Yes. Flow = 3
  - Undo flow

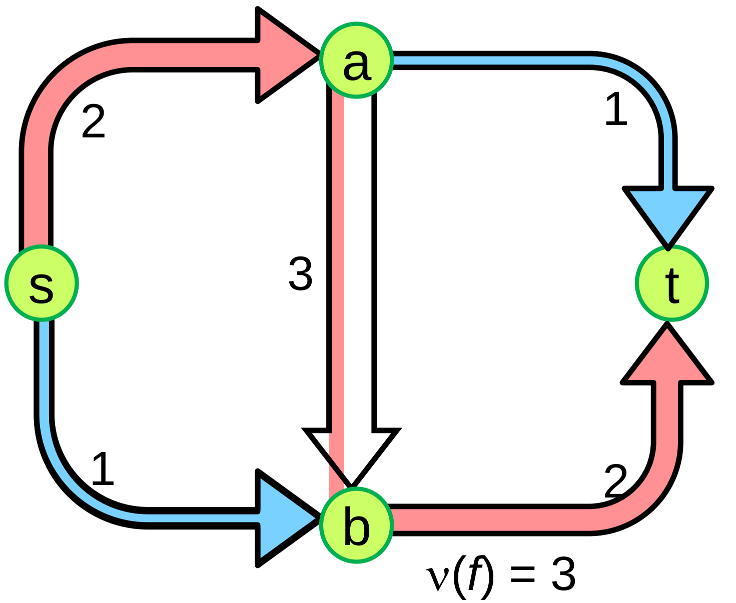


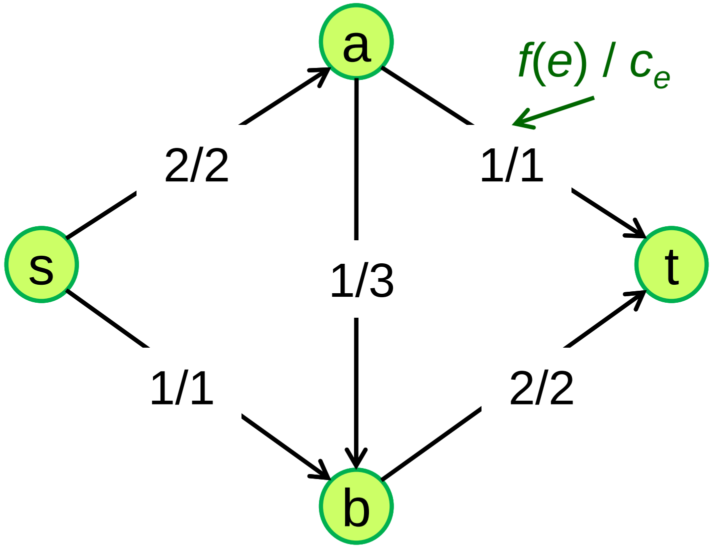


The Maximum Flow Problem

- Given: A flow network
- Goal: Find a max possible flow

- Flow definition:
  - s-t flow f: $E \rightarrow \Re^+$
  - A function f that maps each edge e to a nonnegative real number
    - $f(e)$: flow carried by edge e
  - $\nu(f)$: the value of a flow f
    - $\nu(f) = \sum_{e \text{ out of } s} f(e)$ (flow generated at s)
- Flow properties
  1. Capacity conditions:
     - $\forall e \in E$, $0 \leq f(e) \leq c_e$
  2. Conservation conditions:
     - $\forall u \in V \setminus \{s, t\}$, $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$

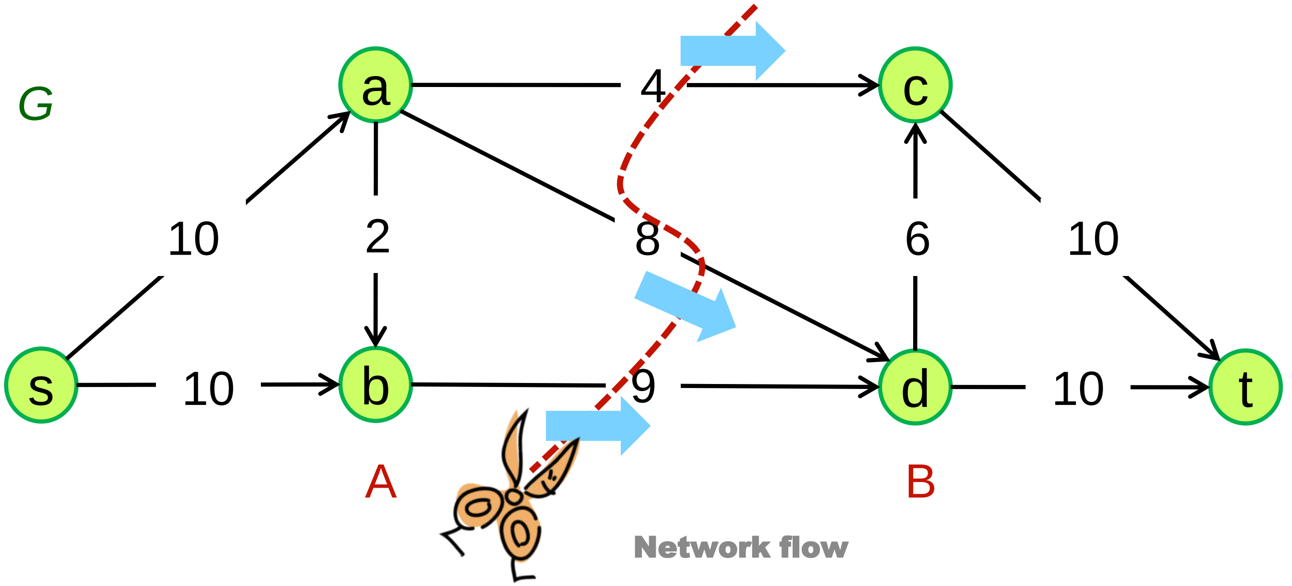




Upper Bounds of the Maximum *s*-*t* Flow

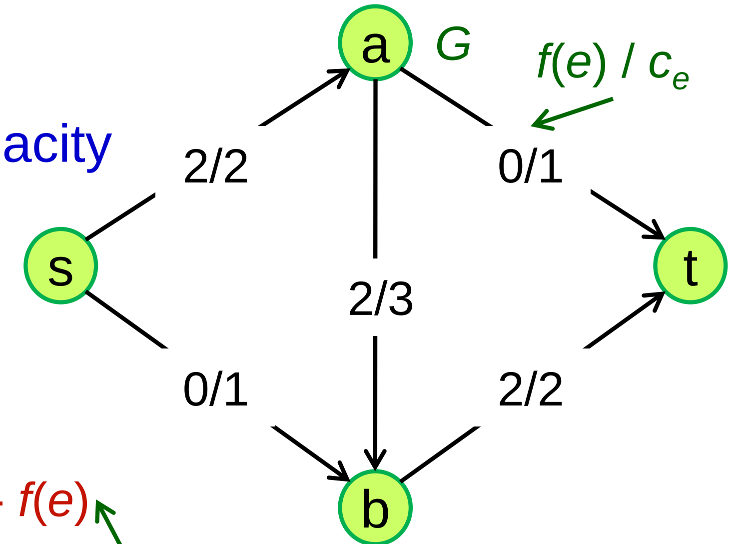
- Q: Can we find the upper bound of the *s*-*t* flow?
- A: Yes!
  - Divide the nodes into two sets, *A* and *B*, so that $s \in A$ and $t \in B$.
  - Any *s*-*t* flow must cross from *A* into *B* at some point.
  - The *s*-*t* flow uses up some of the edge capacity from *A* to *B*.
- Each "cut" places an upper bound on the maximum flow.

  $\Rightarrow$ Find cut of minimum capacity = find maximum flow

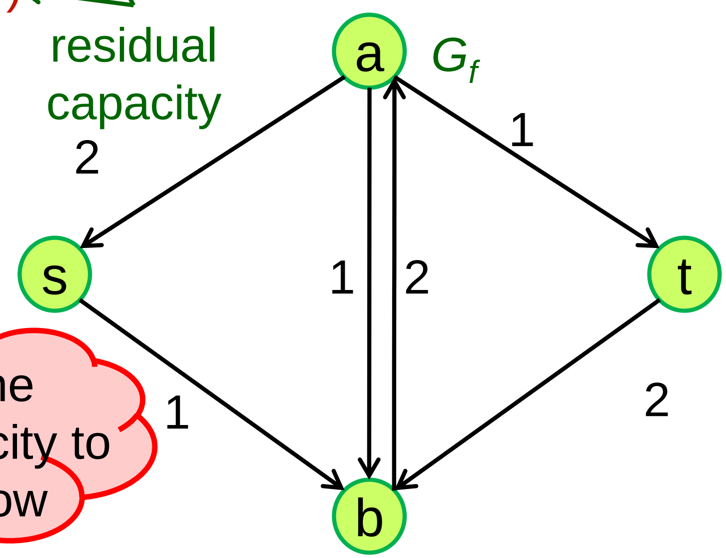


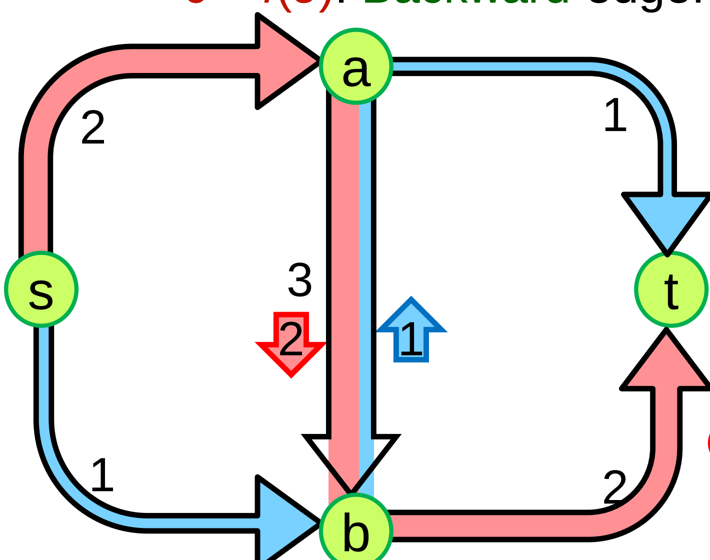
Ford-Fulkerson: Residual Graph

- Pushing flow:
  - Push forward on edges with leftover capacity
  - Push backward on edges with flow
- The residual graph G_f of G w.r.t. f:
  - $V(G_f) = V(G)$
  - For each $e = (u, v) \in E(G)$
    - $f(e) < c_e$: Forward edge: $e' = (u, v)$, $c'_e = c_e - f(e)$
    - $0 < f(e)$: Backward edge: $e'' = (v, u)$, $c''_e = f(e)$



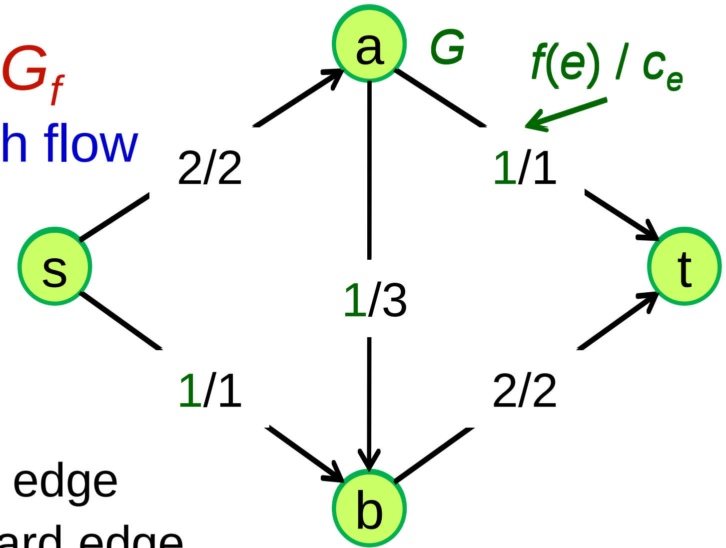
residual capacity



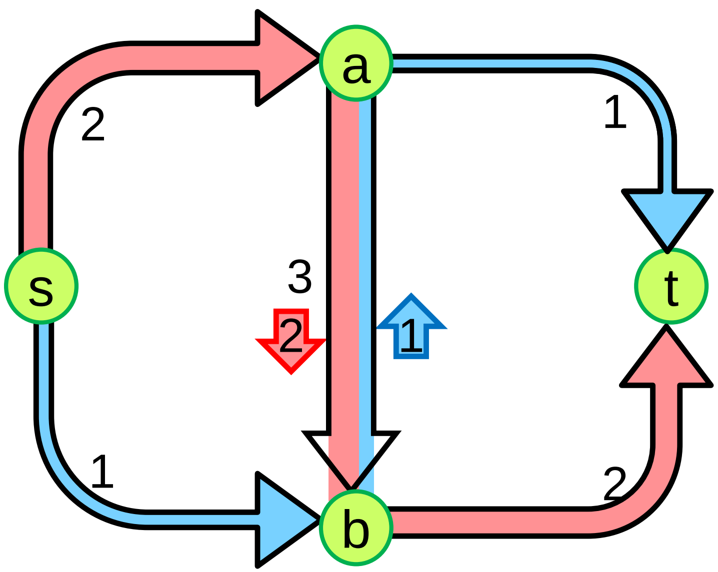


G_f records the remaining capacity to push more flow

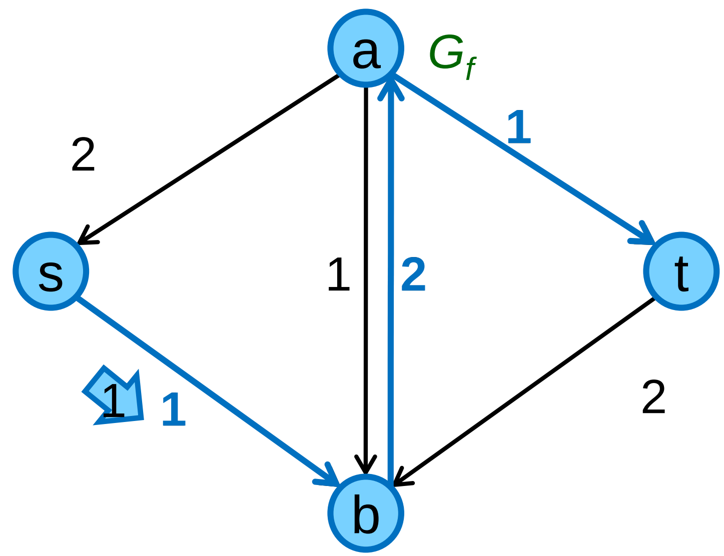
Augmenting Paths in a Residual Graph



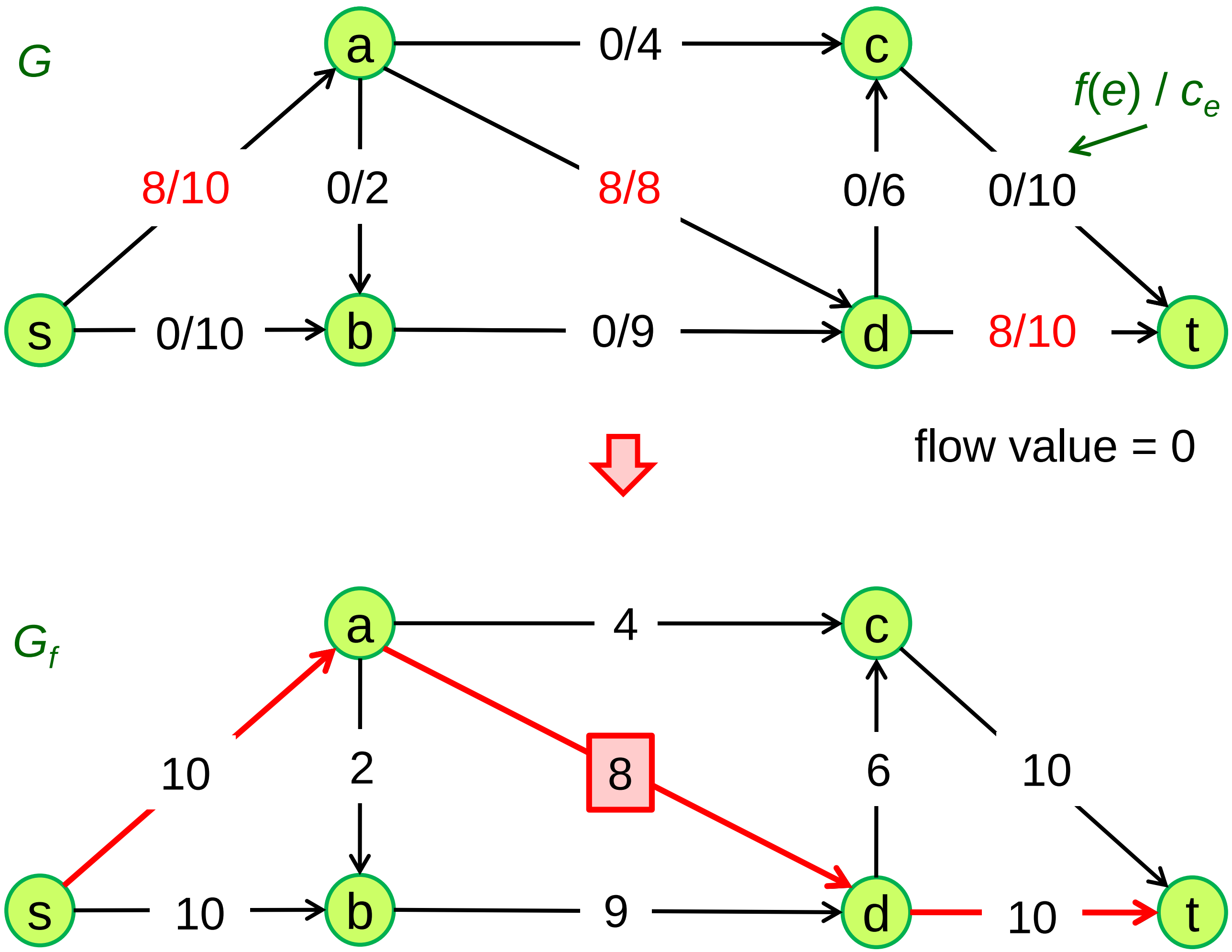
- Pushing flow = augmenting path in G_f
  - G_f records the remaining capacity to push flow
  - Let P be a simple s-t path in G_f
    - bottleneck(P, f) = min res. cap on P
  - Push bottleneck(P, f) units of flow
  - New s-t flow: $\nu(f)$ + bottleneck(P, f)
    - Increase $f(e)$ by bottleneck(P, f) for forward edge
    - Decrease $f(e)$ by bottleneck(P, f) for backward edge



=

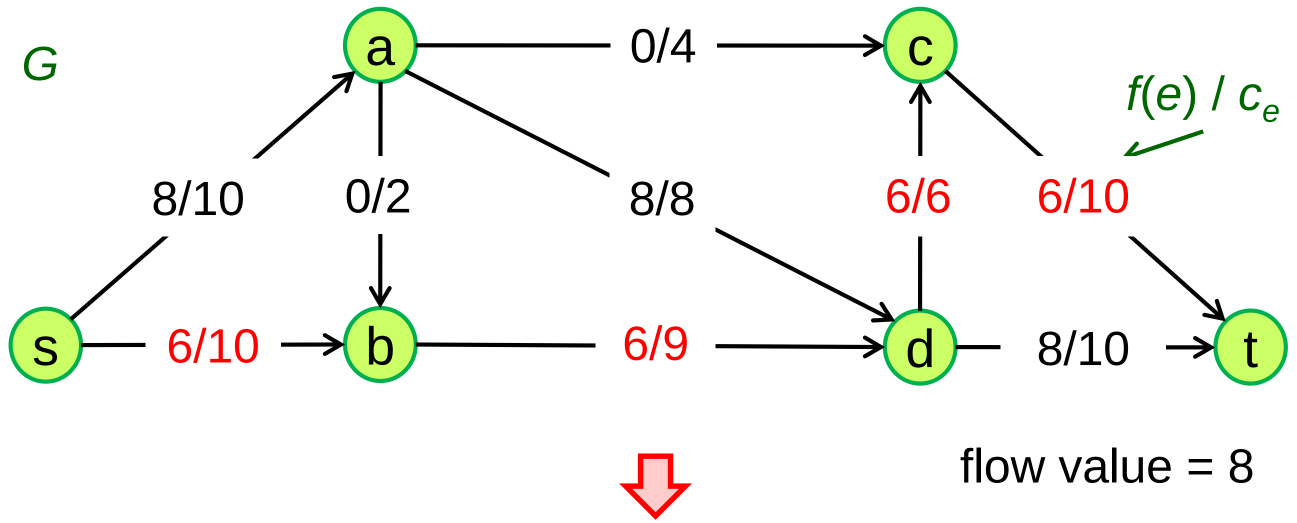


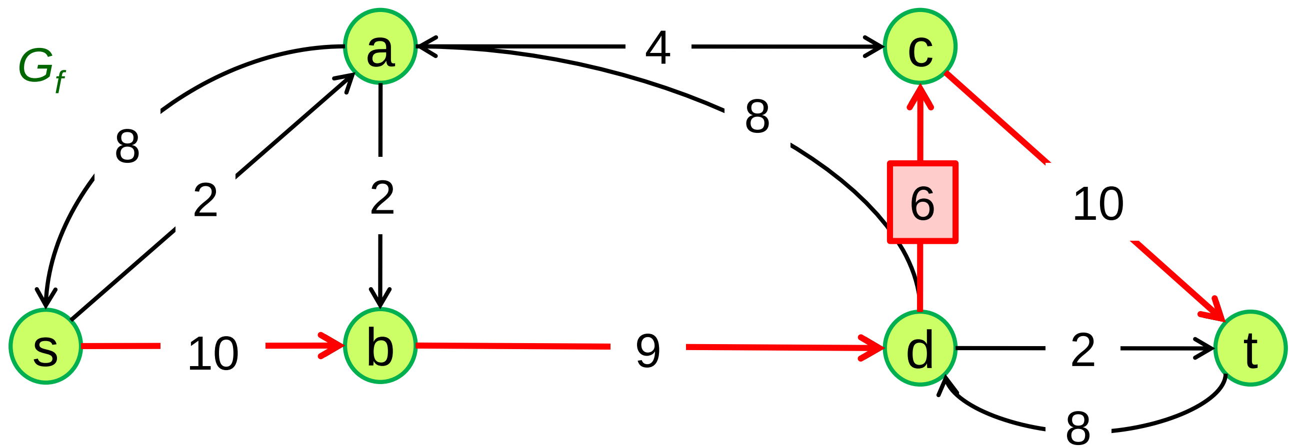
Ford-Fulkerson: Example (1/5)



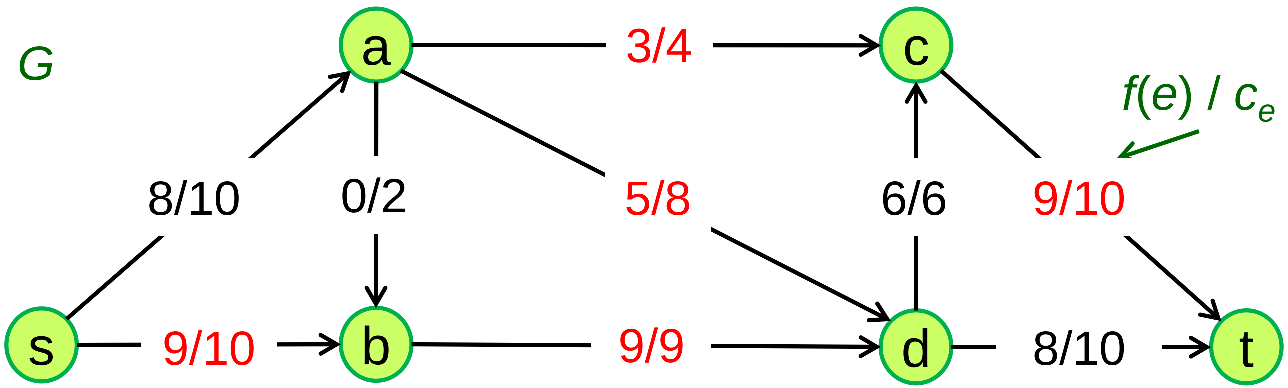
L. R. Ford & D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics* (8) pp.399–404, 1956.

Ford-Fulkerson: Example (2/5)

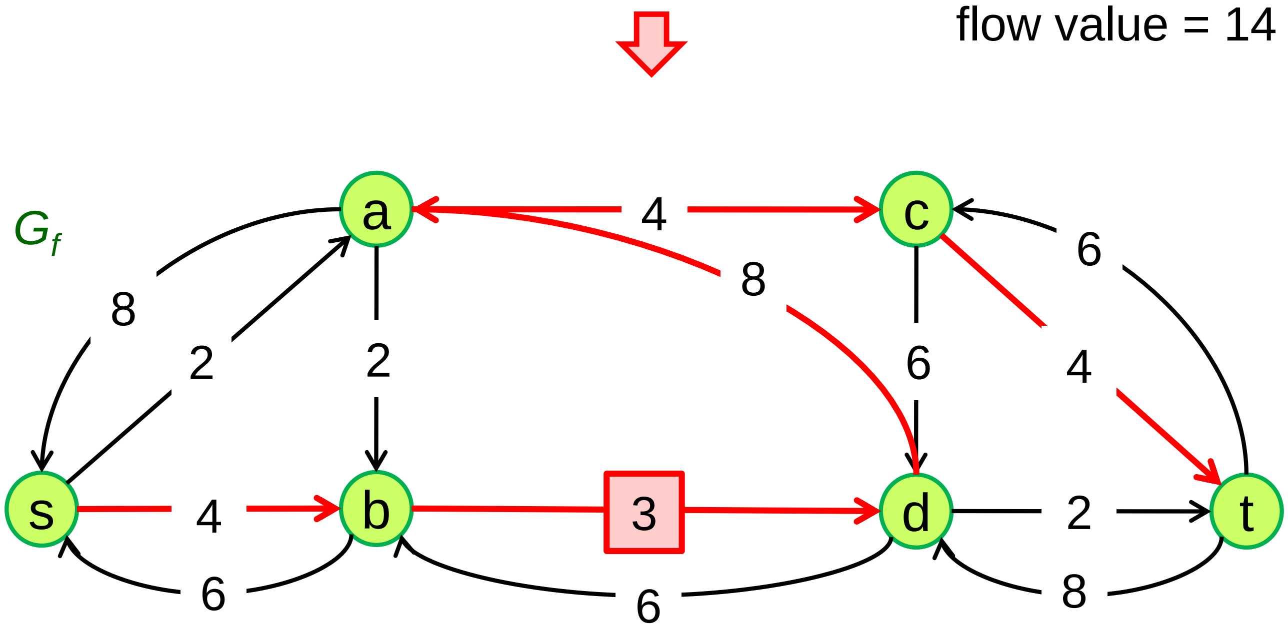




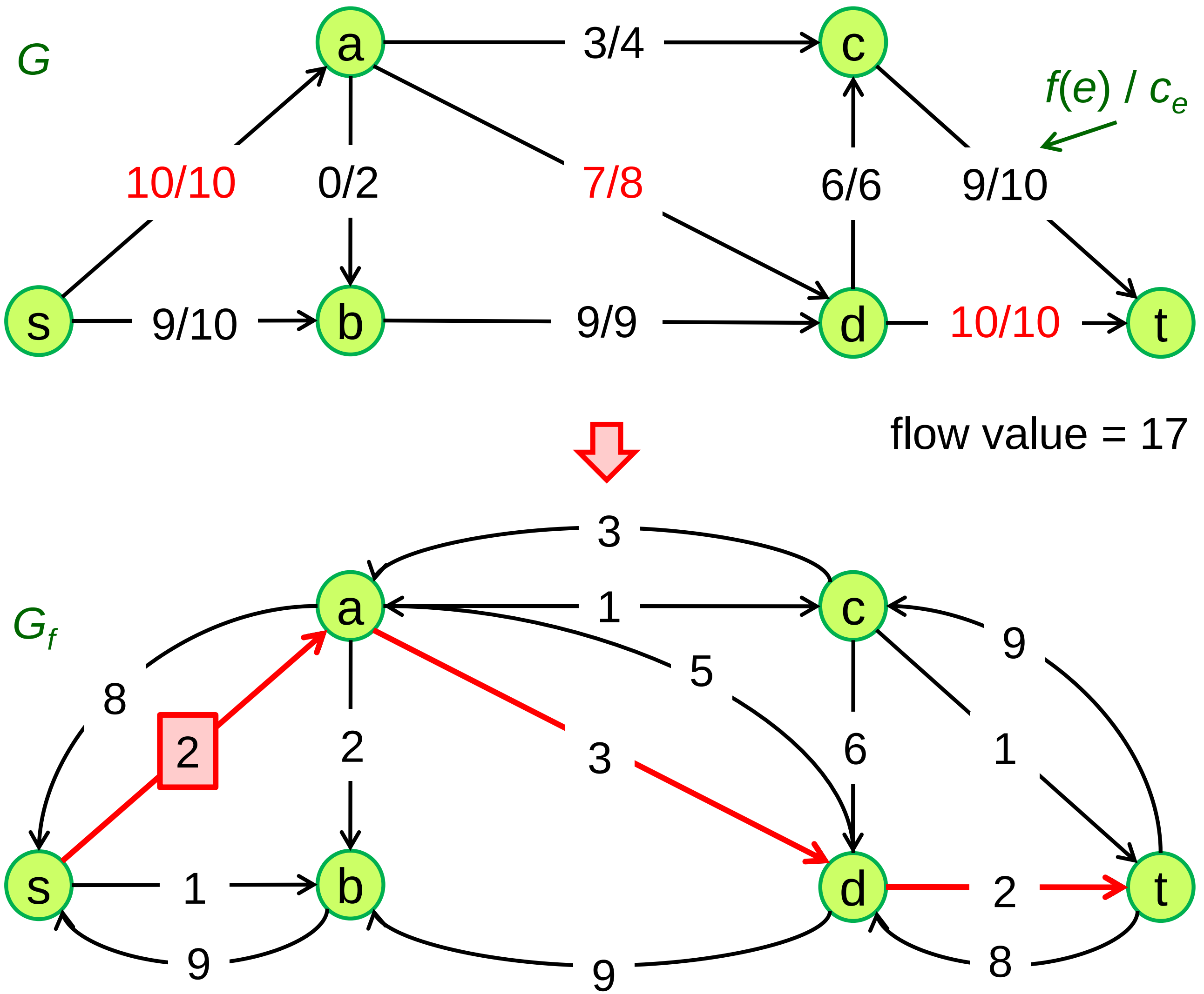
Ford-Fulkerson: Example (3/5)



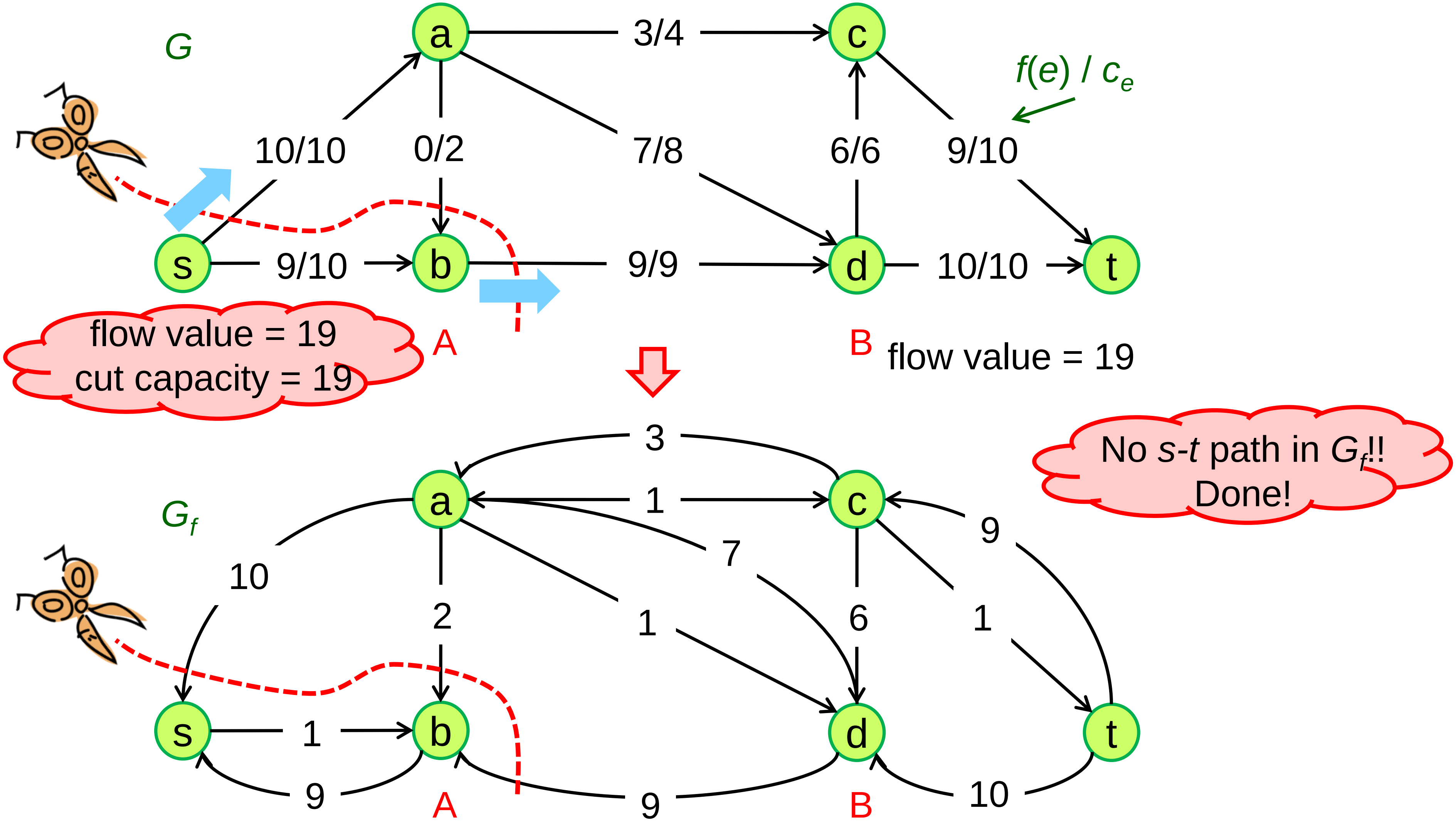
flow value = 14



Ford-Fulkerson: Example (4/5)



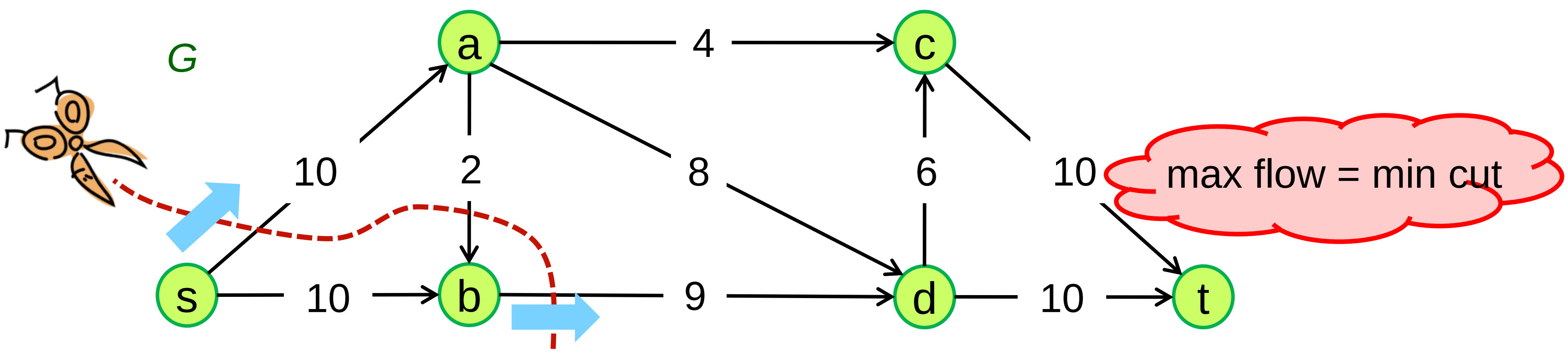
Ford-Fulkerson: Example (5/5)



Ford-Fulkerson Algorithm

Iteratively push flow forward and backward via flow network and residual graph

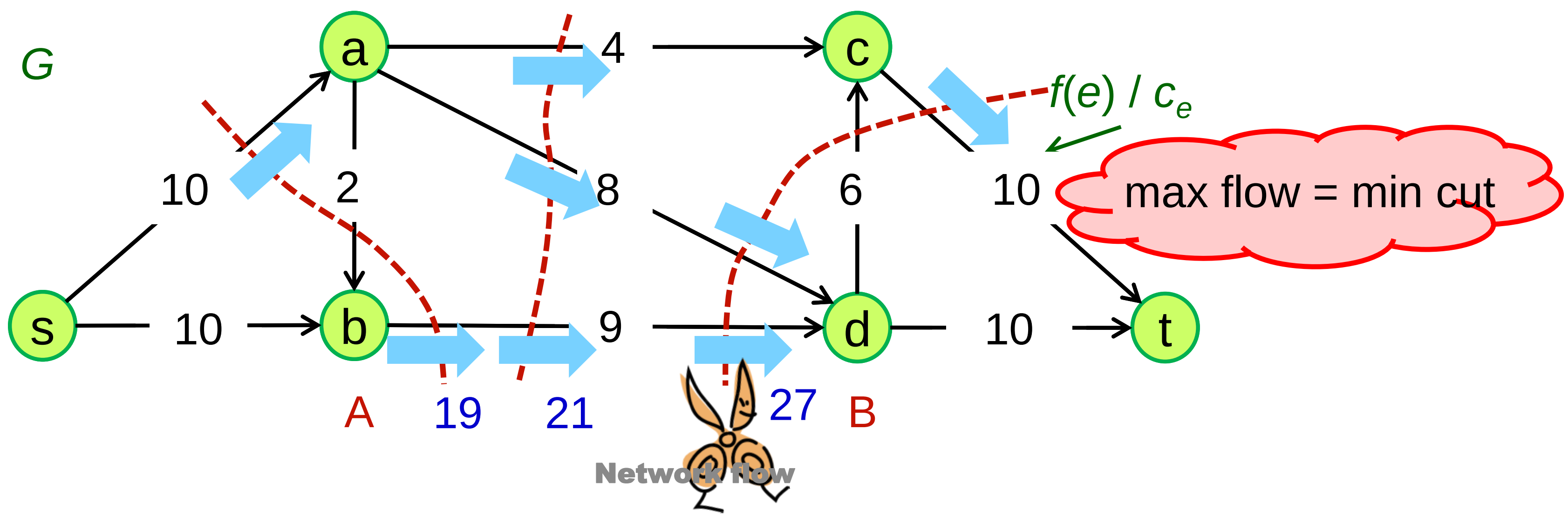
- Procedure
  1. Start with *f*(*e*) = 0 for all edge $e \in E$ and construct the residual graph.
  2. Find an *s*-*t* path *P* in the residual graph.
  3. Augment flow along path *P* and update the residual graph.
  4. Repeat steps 2-3 until you get stuck.
- Optimal



L. R. Ford & D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics* (8) pp.399–404, 1956.

The Max-Flow Min-Cut Theorem

- The value of max flow is equal to the value of the min cut.
  - Cut: Divide *V* into two disjoint sets, *A* and *B*, s.t. $s \in A$ and $t \in B$.
  - Min-cut: minimize the sum of the capacities of the cut edges directed from *A* to *B*
- Observation:
  - Any *s*-*t* flow must cross from *A* into *B* at some point and uses up some of the capacities of the cut edges.
  - Each cut places an upper bound on the max value of an *s*-*t* flow.



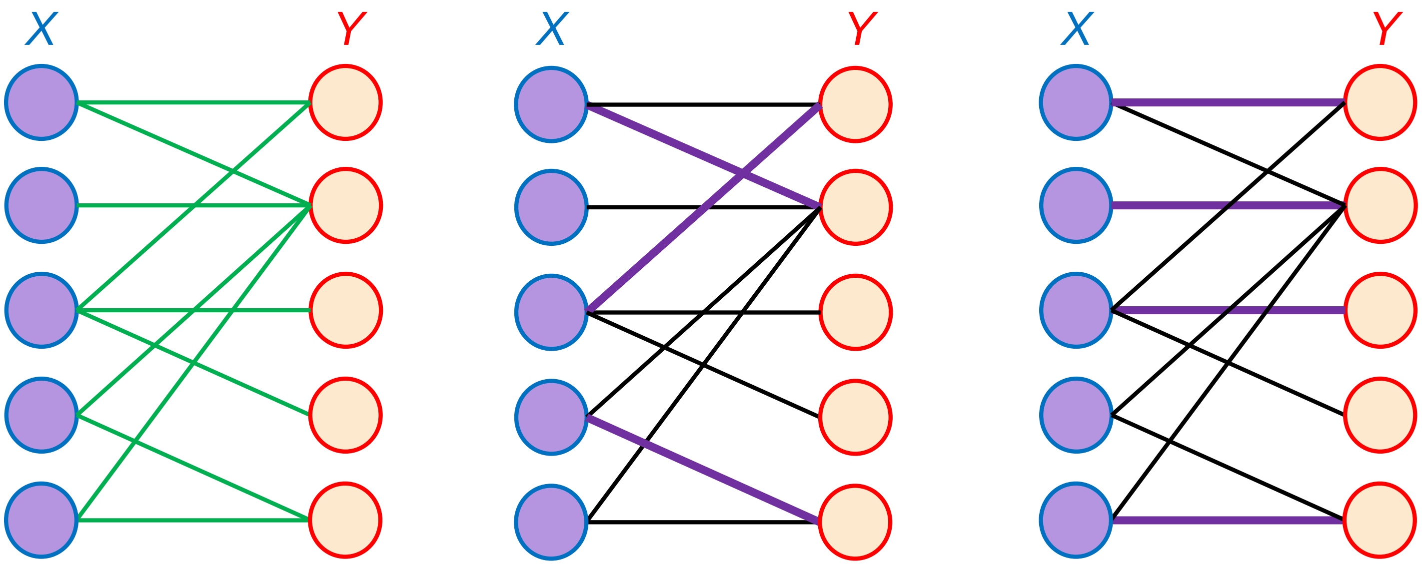
Bipartite Matching

If You Were a Matchmaker…

- Imagine you are a matchmaker
  - One hundred female clients, and one hundred male clients
  - Each woman has compiled a list of her prince charming criteria
  - Each man has compiled a list of her princess snow white criteria
  - Your job is to arrange one hundred suitable marriages
    - Neither singlehood nor polygamy
    - (The more marriages, the more money)

Bipartite Matching

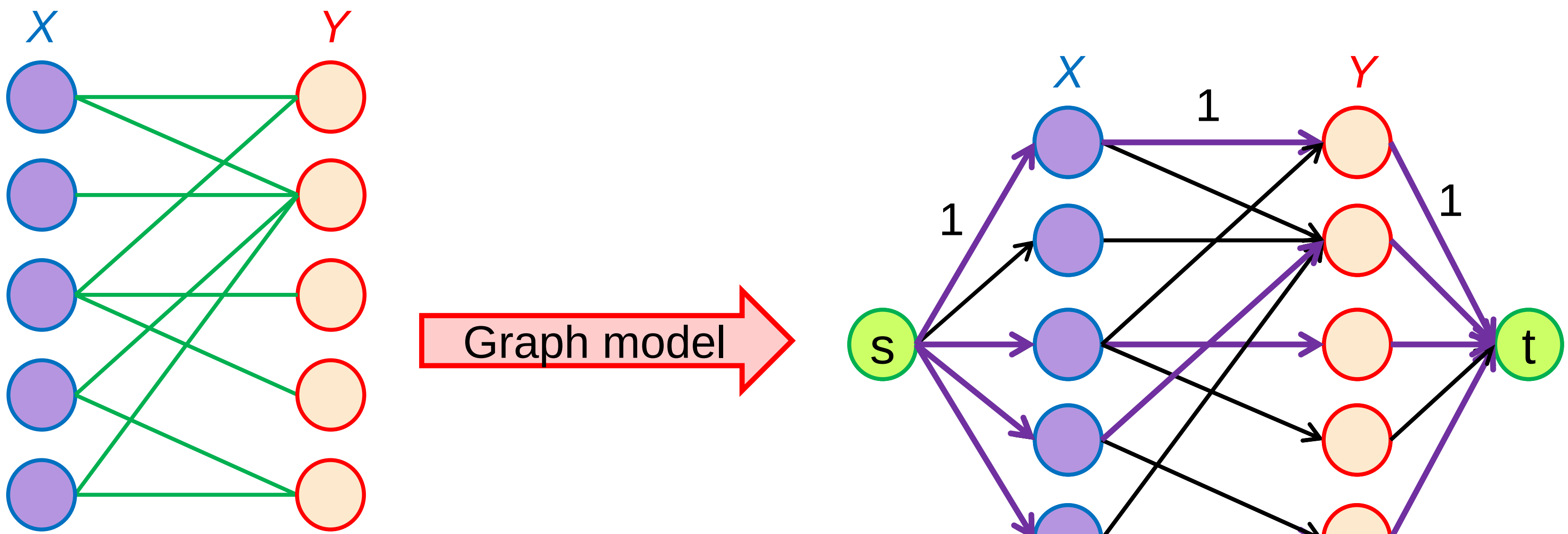
- Given: *n* men, *m* women and their feasible partners, $G = (X, Y, E)$
- Goal: Find the matching $M \subseteq E$ with the max # of marriages
- Perfect matching: Everyone is matched monogamously.
  - Each man gets exactly one woman
  - Each woman gets exactly one man



$G = (X, Y, E)$ matching: 3 max matching: 4

Bipartite Matching via Network Flow (1/2)

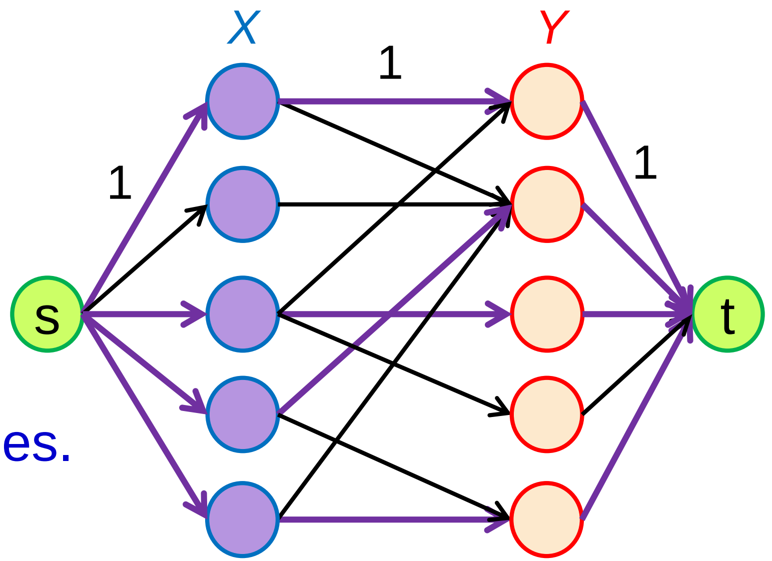
- Bipartite matching can be solved via reduction to max flow.
  - All edges are directed from *X* to *Y.*
  - Add nodes *s* and *t*
  - Add edges (*s*, *x*) for all $x \in X$, (*y*, *t*) for all $y \in Y$
  - All capacities are set to 1.
  - Flow corresponds to matched pairs.

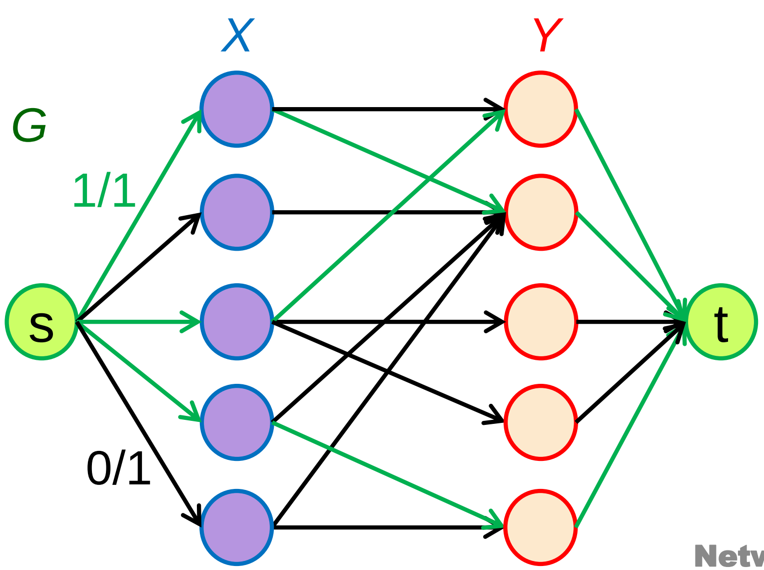


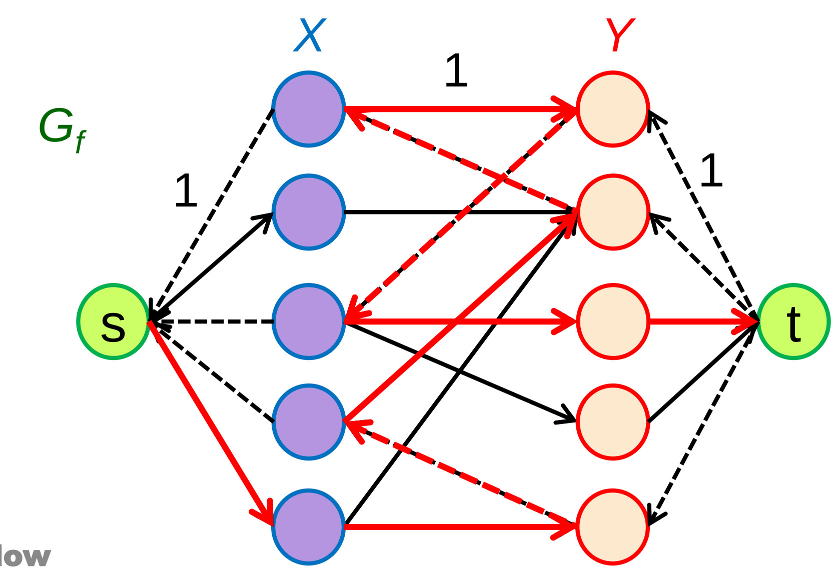
$G = (X, Y, E)$

Bipartite Matching via Network Flow (2/2)

- Residual graph G_f simplifies to:
  - If $(x, y) \notin M$, then (x, y) is in G_f.
  - If $(x, y) \in M$, the (y, x) is in G_f.
- Augmenting path simplifies to:
  - Edge from s to an unmatched $x \in X$.
  - Alternating unmatched/matched edges.
  - Edge from unmatched $y \in Y$ to t.





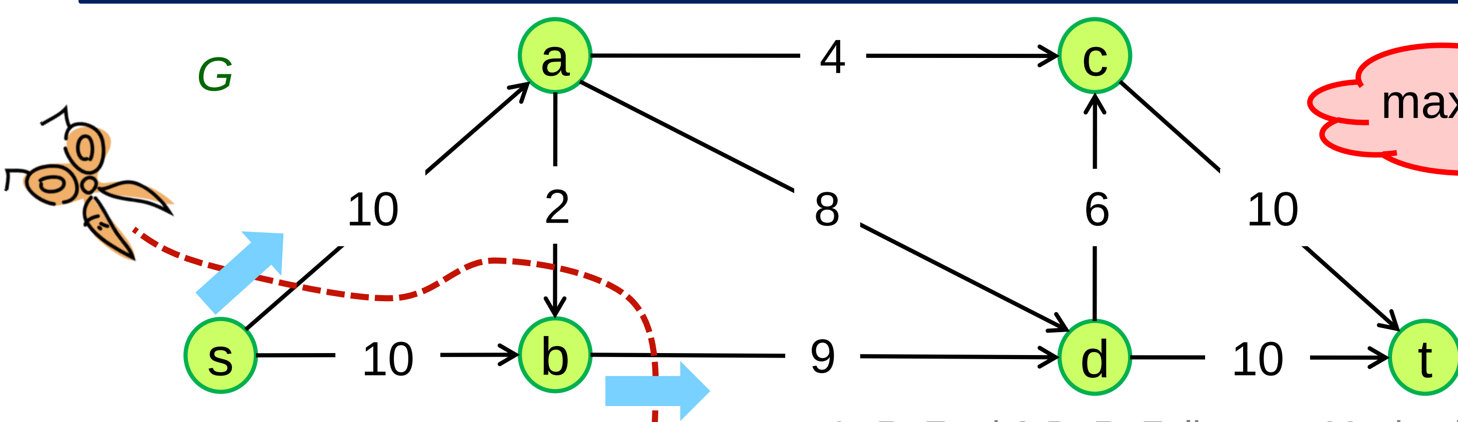


Maximum Flows and Minimum Cuts

Ford-Fulkerson Algorithm

Iteratively push flow forward and backward via flow network and residual graph

```
Ford-Fulkerson(G, s, t)
foreach (e ∈ E) do
    f(e) = 0
construct G_f
while (∃ an s-t path P in G_f) do          // augmenting path
    b = bottleneck(P, f)
    foreach (e ∈ P) do
        if (e ∈ E) then f(e) = f(e) + b     // forward edge
        else f(e^R) = f(e^R) – b            // backward edge
    update G_f
return f
```

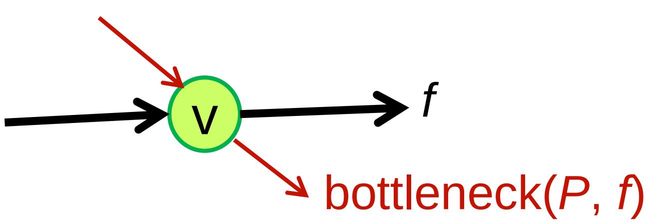


max flow = min cut

L. R. Ford & D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics* (8) pp.399–404, 1956.

Is It a Flow?

- Let f' be the new flow after line 8 in Ford-Fulkerson algorithm. f' is a flow in G.
- Pf: Verify the capacity and conservation conditions.
  - f' differs from f only on edges of P in G (a s-t simple path in G_f).
  - Capacity condition: (check edges of P in G)
    - If $e = (u, v) \in P$ is a forward edge, its residual capacity $= c_e - f(e)$.
      $0 \leq f(e) \leq f'(e) = f(e) + \text{bottleneck}(P, f) \leq f(e) + (c_e - f(e)) = c_e$
    - If $(u, v) \in P$ is a backward edge, its residual capacity is $f(e)$, $e = (v, u)$
      $c_e \geq f(e) \geq f'(e) = f(e) - \text{bottleneck}(P, f) \geq f(e) - f(e) = 0$
  - Conservation condition: (check nodes of P in G)
    - For each internal node v on P, the amount of flow belonging to f satisfies the conservation condition.
    - Excluding f, f' enters and exits v with bottleneck(P, f)
    - f' must satisfy conservation condition, too.



Termination and Running Time (1/2)

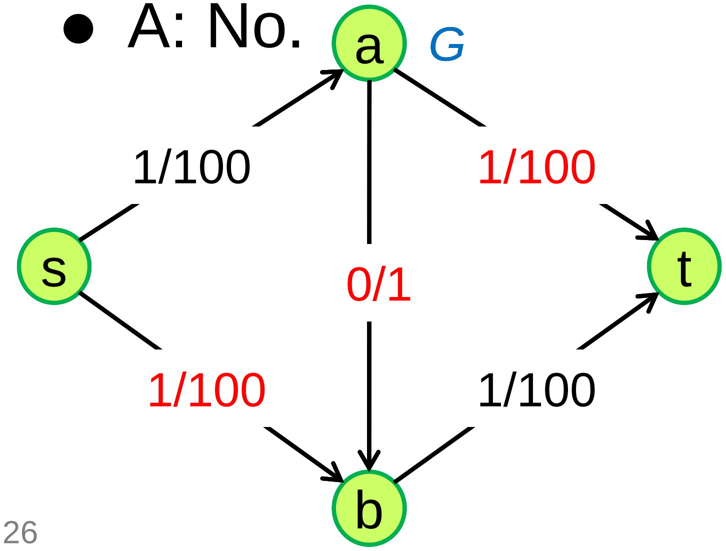
- Assumption: all capacities are integers.
- Invariant property: Throughout Ford-Fulkerson, the flow values $\{f(e): e \in E\}$ and the residual capacities in G_f are integers.

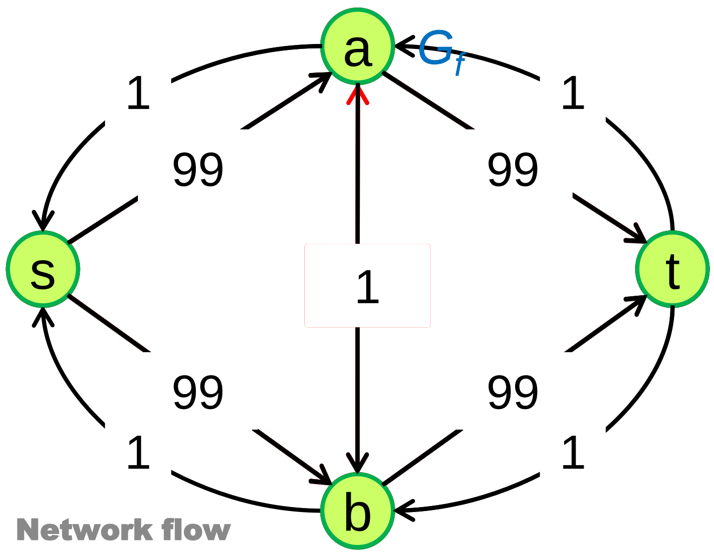
The flow value strictly increases when we apply an augmentation

- $\nu(f') = \nu(f)$ + bottleneck(P, f). Since bottleneck(P, f)>0, $\nu(f') > \nu(f)$.
- Pf:
  - The first edge e of P must be an edge out of s in G_f.
  - Since P is simple, it does not visit s again. Since G has no edges entering s, e must be a forward edge.
  - We increase $f(e)$ by bottleneck(P, f) > 0, and we do not change the flow on any other edge out of s.
  - Therefore, $\nu(f') = \nu(f)$ + bottleneck(P, f). $\nu(f') > \nu(f)$.

Termination and Running Time (2/2)

- $C = \sum_{e \text{ out of } s} c_e \geq \sum_{e \text{ out of } s} f(e) = \nu(f)$. Ford-Fulkerson terminates in at most C iterations of the while loop.
- Pf: Each augmentation increases flow value by at least 1.
- Running time: $O(mC)$
  - while loop: C iterations
  - Augmentation: $O(m)$
- Q: Is Ford-Fulkerson polynomial in input size?
- A: No.

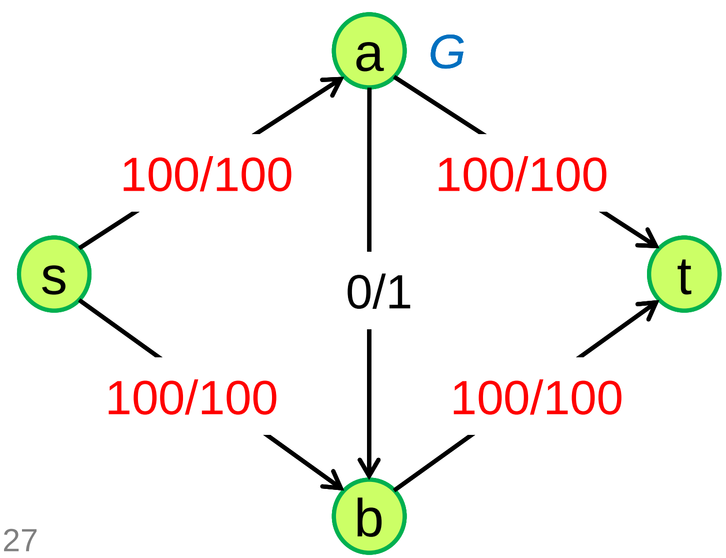


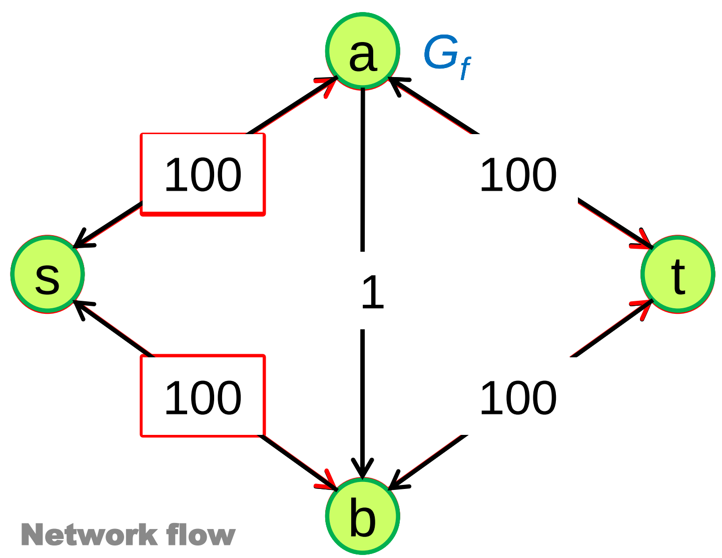


Choosing Good Augmenting Paths

- Use care when selecting augmenting paths.

- Choose augmenting paths with:
  - Max bottleneck capacity
  - Fewest number of edges
  - Sufficiently large bottleneck capacity
    - △-scaling: look for paths with bottleneck capacity of at least △





Max-Flow Min-Cut Theorem (1/3)

- Max-flow min-cut theorem: The value of the max flow is equal to the value of the min cut.
- Pf: We prove both simultaneously by showing:
  1. There exists a cut (A, B) such that $\nu(f) = \text{cap}(A, B)$.
  2. Flow f is a max flow.
  3. There is no augmenting path relative to f.

- 2. $\Rightarrow$ 3. By contradiction.
  - Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along path.

Max-Flow Min-Cut Theorem (2/3)

- Weak duality. Let f be any flow. Then, for any s-t cut (A, B) we have $\nu(f) \leq \text{cap}(A, B)$.
- Pf:
  - $\nu(f) = \sum_{e \text{ out of } s} f(e)$

    $= \sum_{e \text{ out of } s} f(e) - \cancelto{0}{\sum_{e \text{ into } s} f(e)}$

    $= \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e) + \sum_{v \in A} (\cancel{\sum_{e \text{ out of } v} f(e)} - \cancel{\sum_{e \text{ into } v} f(e)})$

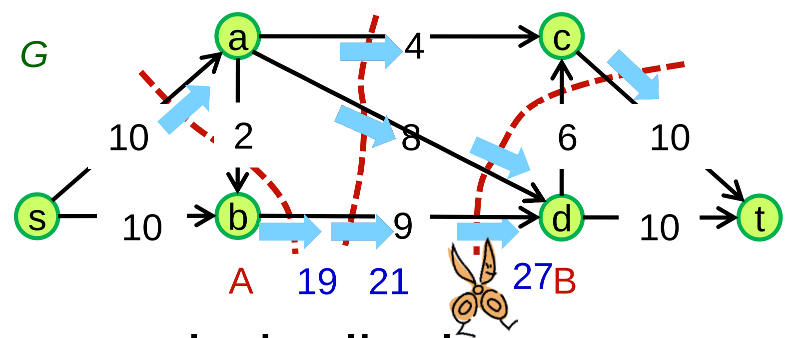
    (internal edges in A: $f(e)$ appears once "+" & once "-")

    $= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$

    $\leq \sum_{e \text{ out of } A} f(e)$

    $\leq \sum_{e \text{ out of } A} c_e$

    $= \text{cap}(A, B)$.



- 1. $\Rightarrow$ 2. This was the corollary to weak duality lemma.
  - All edges into A are completely unused.

Max-Flow Min-Cut Theorem (3/3)

- 3. $\Rightarrow$ 1.
  - Let f be a flow without augmenting paths.
  - Let A be set of nodes reachable from s in residual graph.
  - By definition of A, $s \in A$.
  - By definition of f, $t \notin A$.
  - $\nu(f) = \sum_{e \text{ out of } s} f(e)$

    $= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$

    $= \sum_{e \text{ out of } A} f(e)$ // $f(e)_{\text{for } e \text{ into } A} = 0$, otherwise, s can reach out

    $= \sum_{e \text{ out of } A} c_e$

    $= \text{cap}(A, B)$.