CSE 202: Design and Analysis of Algorithms

Lecture 8

Next: Network Flows

Oil Through Pipelines

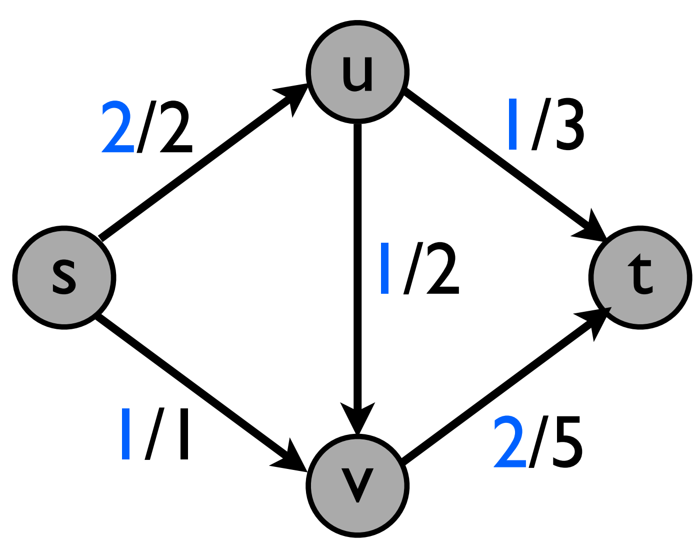
Problem: Given directed graph G=(V,E), source s, sink t, edge capacities c(e), how much oil can we ship from s to t?

An s-t flow is a function: E $\longrightarrow$R such that:

- 0 <= f(e) <= c(e), for all edges e
- flow into node v = flow out of node v, for all nodes v except s and t,

$$\sum_{e \; into \; v} f(e) = \sum_{e \; out \; of \; v} f(e)$$

Size of flow f = Total flow out of s = total flow into t

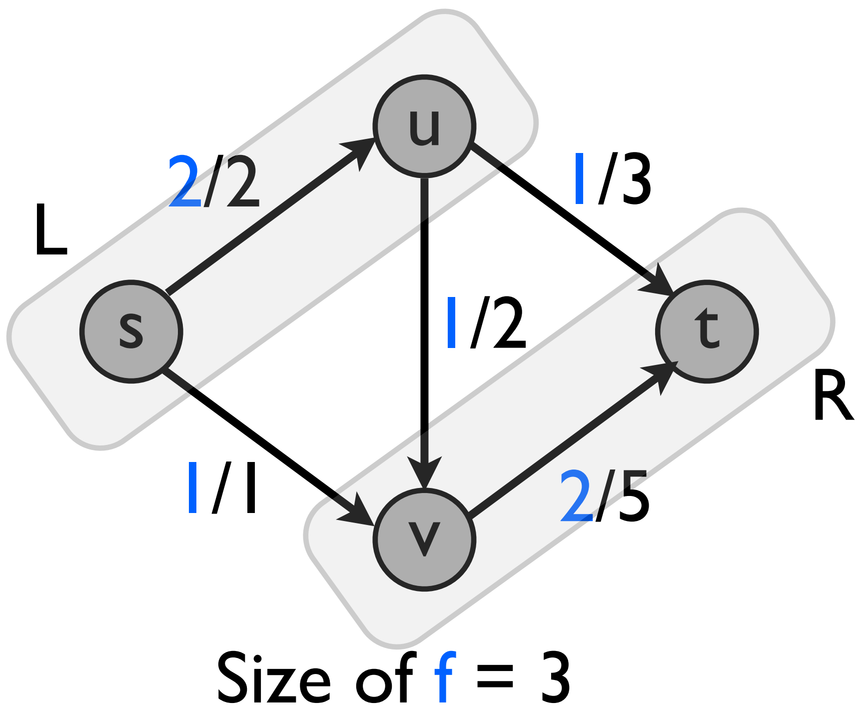


Size of f = 3

The Max Flow Problem: Given directed graph G=(V,E), source s, sink t, edge capacities c(e), find an s-t flow of **maximum size**

Flows and Cuts

The Max Flow Problem: Given directed graph G=(V,E), source s, sink t, edge capacities c(e), find an s-t flow of maximum size



An **s-t Cut** partitions nodes into groups = (L, R) s.t. s in L, t in R

Capacity of a cut (L, R) = $\sum_{(u,v)\in E, u\in L, v\in R} c(u,v)$

Flow across (L,R) = $\sum_{(u,v)\in E, u\in L, v\in R} f(u,v) - \sum_{(v,u)\in E, u\in L, v\in R} f(v,u)$

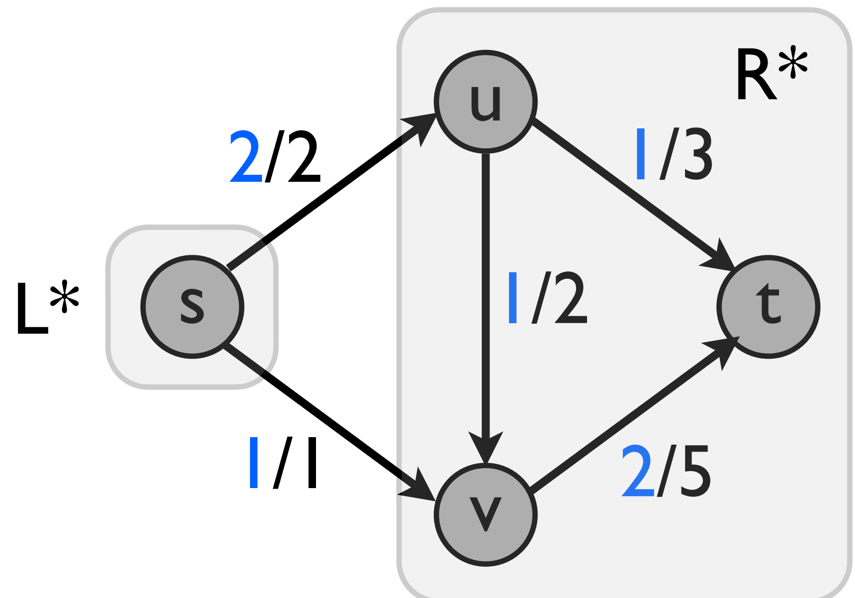
Property: For any flow f, any s-t cut (L, R), size(f) <= capacity(L, R)

Proof: For any cut (L,R), Flow Across (L,R) cannot exceed capacity(L,R)
From flow conservation constraints, size(f) = flow across(L,R) <= capacity(L,R)

Max-Flow <= Min-Cut

Flows and Cuts

The Max Flow Problem: Given directed graph G=(V,E), source s, sink t, edge capacities c(e), find an s-t flow of maximum size



Size of f = 3

An **s-t Cut** partitions nodes into groups = (L, R) s.t. s in L, t in R

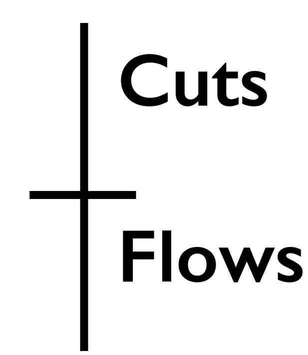
Capacity of a cut (L, R) = $\sum_{(u,v)\in E, u\in L, v\in R} c(u,v)$

Flow across (L,R) = $\sum_{(u,v)\in E, u\in L, v\in R} f(u,v) - \sum_{(v,u)\in E, u\in L, v\in R} f(v,u)$

Property: For any flow f, any s-t cut (L, R), size(f) <= capacity(L, R)

Proof: For any cut (L,R), Flow Across (L,R) cannot exceed capacity(L,R)
From flow conservation constraints, size(f) = flow across(L,R) <= capacity(L,R)

Max-Flow <= Min-Cut



In our example: Size of f = 3, Capacity of Cut (s,V - s) = 3.
Thus, a Min Cut is a **certificate of optimality for a flow**

Ford-Fulkerson algorithm

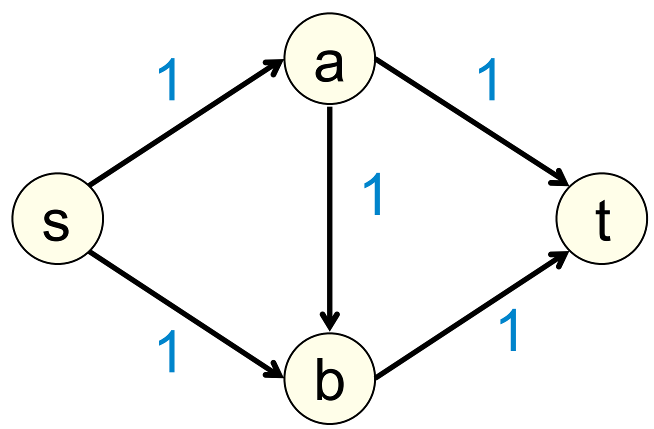
FF Algorithm: Start with zero flow

Repeat:

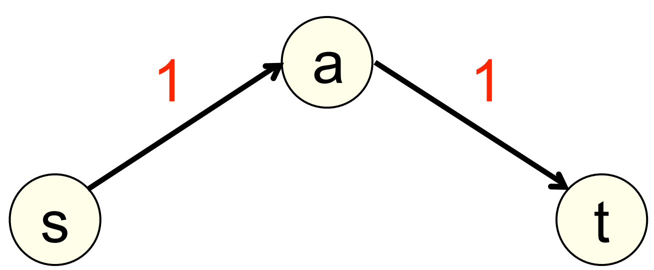
  Find a path from s to t along which flow can be increased

  Increase the flow along that path

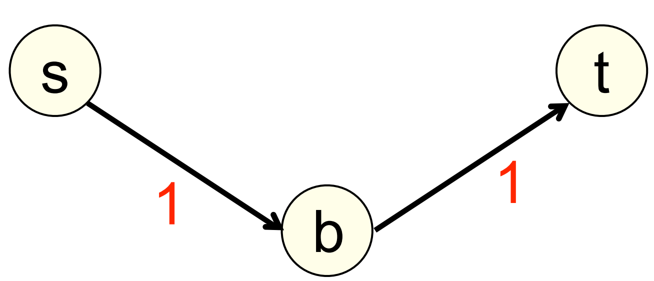
Example



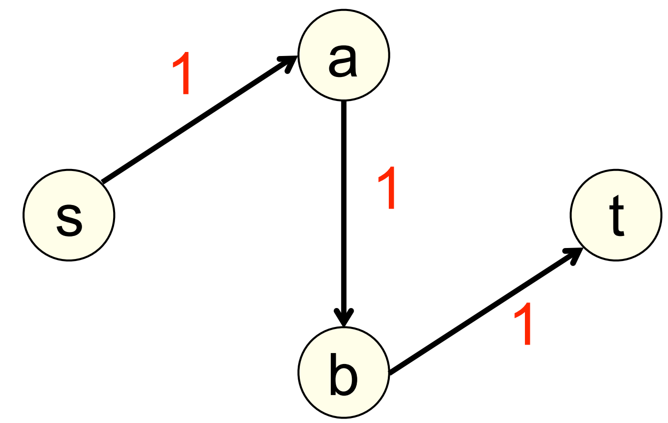
First choose:



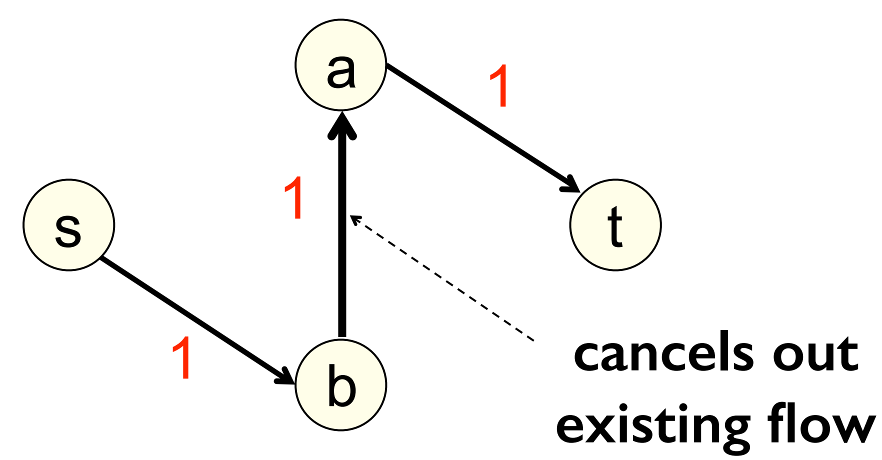
Next choose:



But what if we first chose:



Then we'd have to allow:



Ford-Fulkerson, continued

FF Algorithm: Start with zero flow

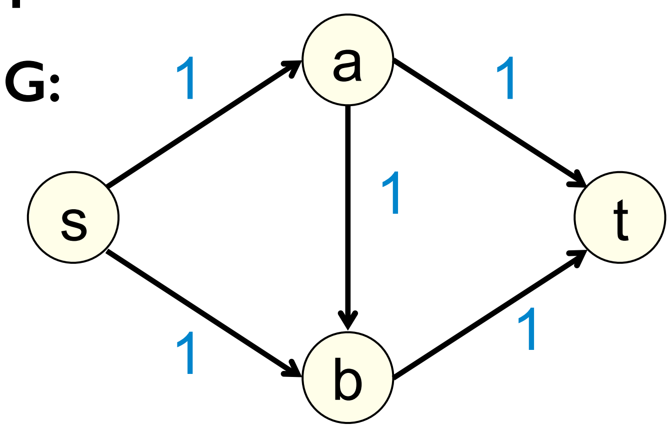
Repeat:

Find a path from s to t along which flow can be increased

Increase the flow along that path

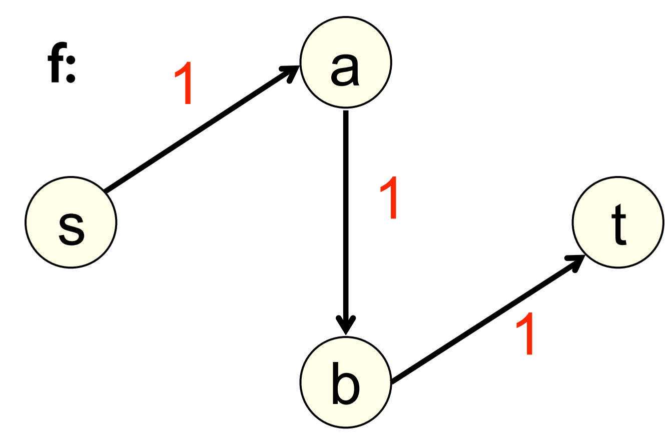
Example

G:



In any iteration, we have some flow f and we are trying to improve it. How to do this?

f:



1: Construct a residual graph G_f ("what's left to take?")

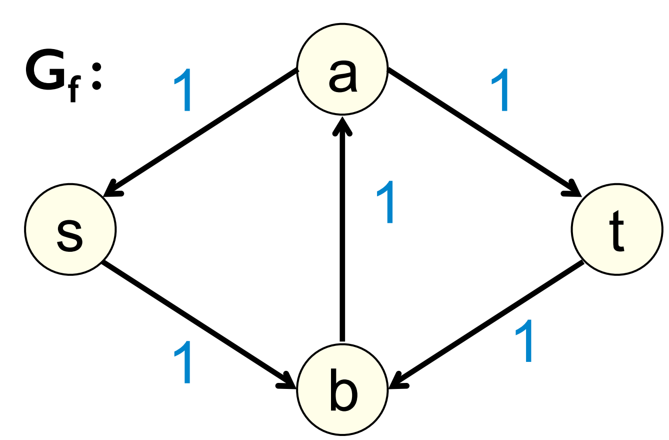
$G_f = (V, E_f)$ where $E_f \subseteq E \cup E^R$

For any (u,v) in E, $c_f(u,v) = c(u,v) - f(u,v)$

any (u,v) in E^R, $c_f(u,v) = f(v,u)$

[ignore edges with zero c_f: don't put them in E_f]

G_f :



2: Find a path from s to t in G_f

3: Increase flow along this path, as much as possible

Example: Round 1

Construct residual graph $G_f = (V, E_f)$

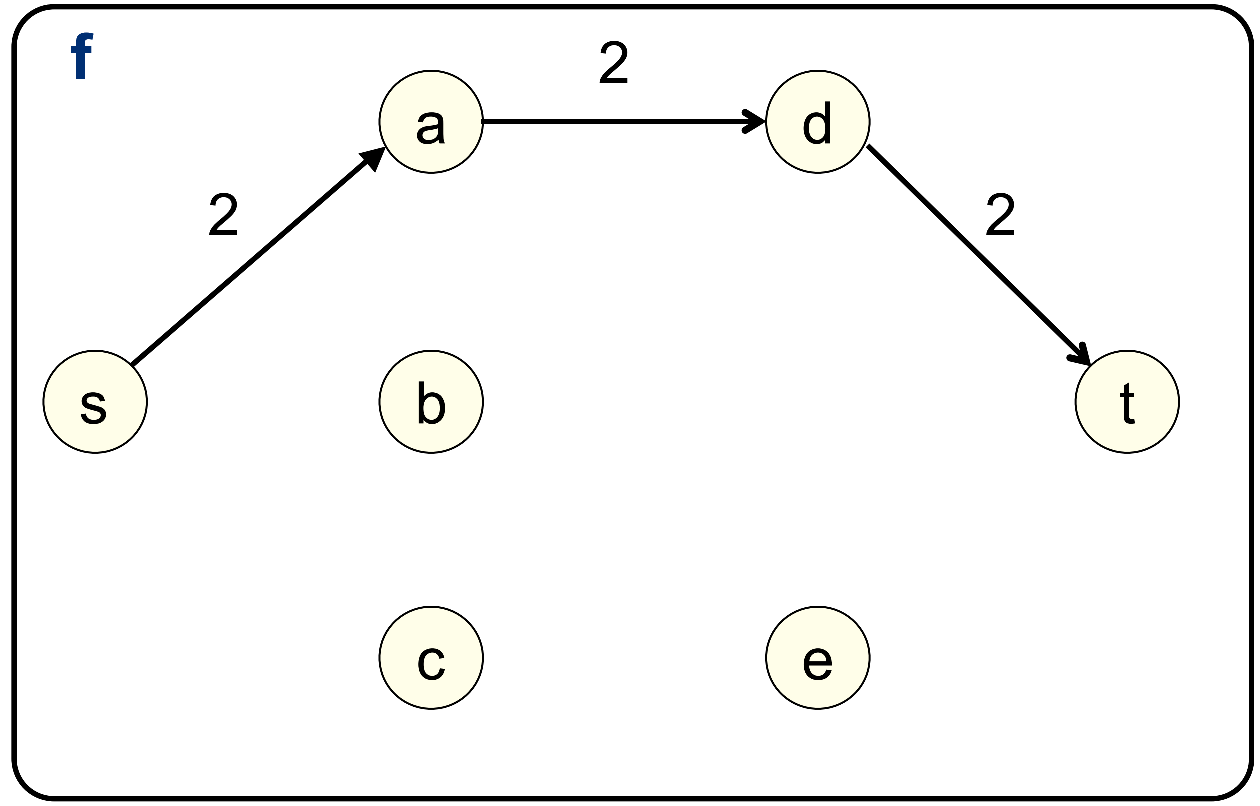
$E_f \subseteq E \cup E^R$

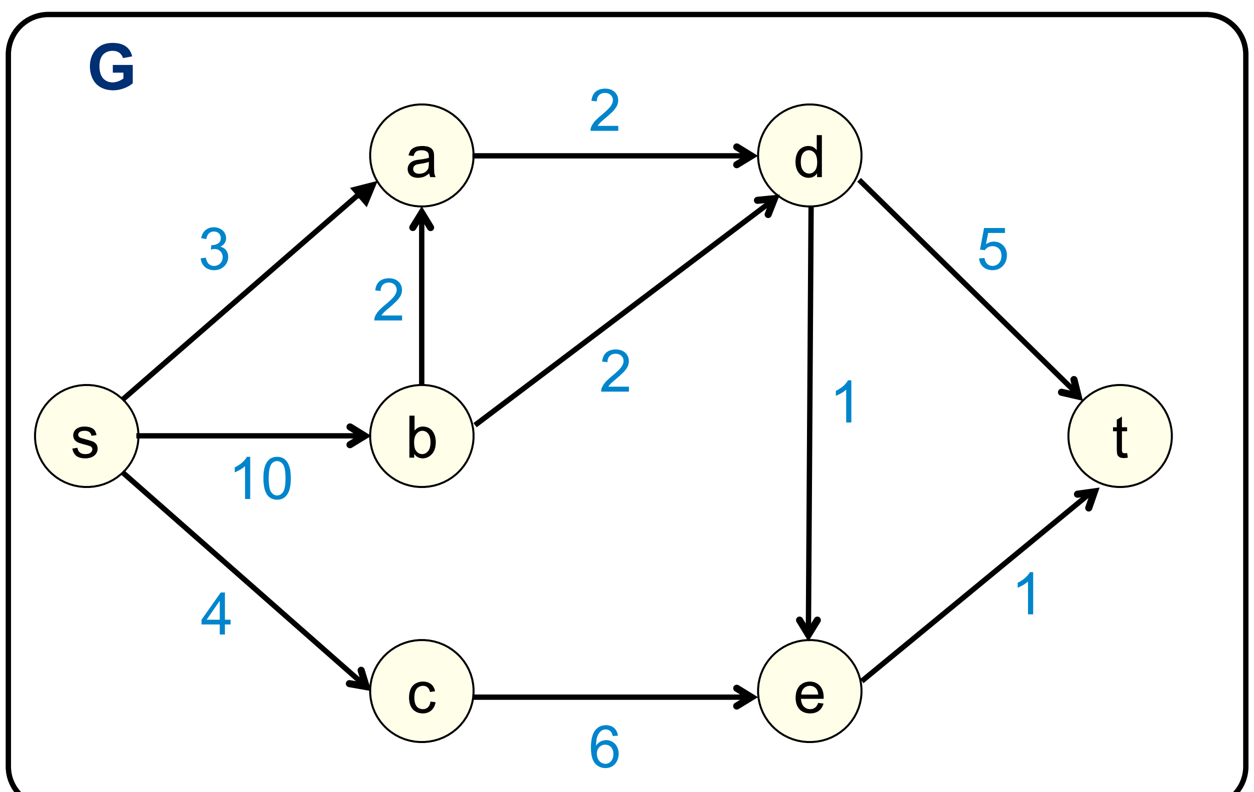
For any (u,v) in E or E^R,

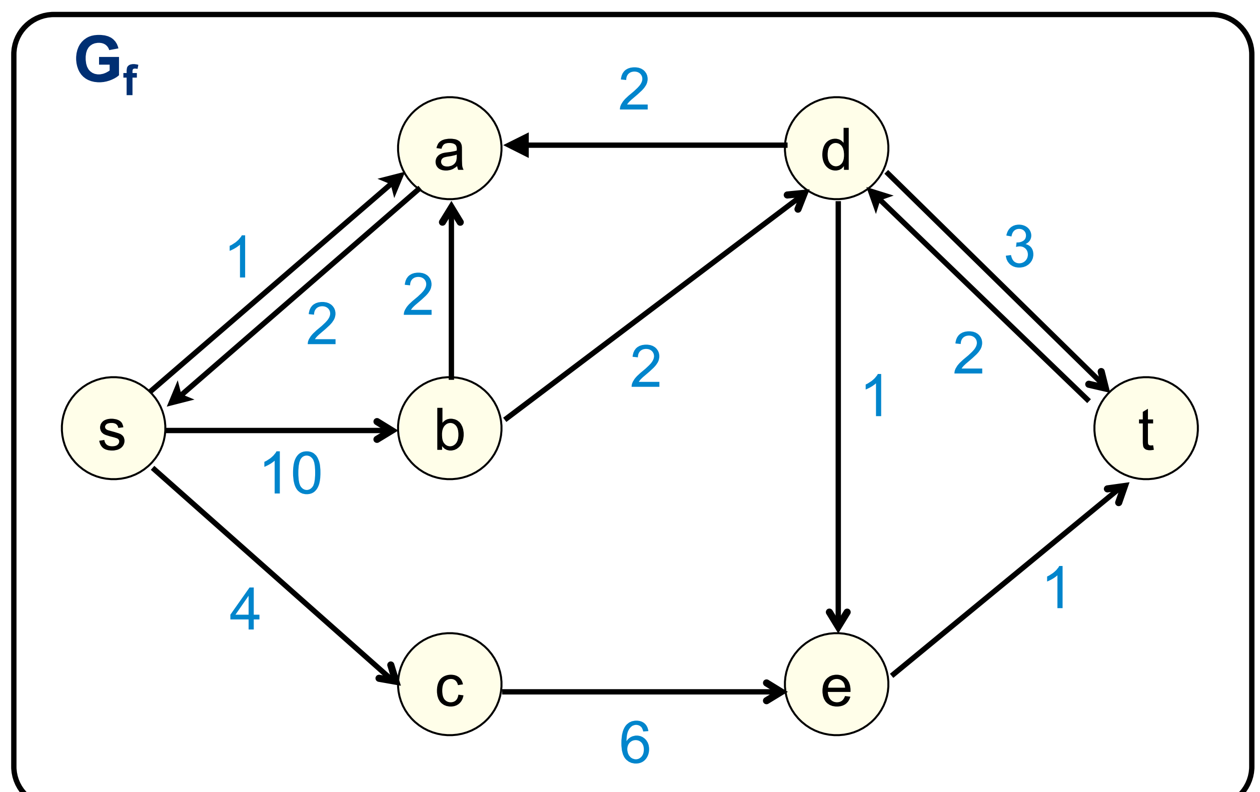
$c_f(u,v) = c(u,v) - f(u,v) + f(v,u)$

Find a path from s to t in G_f

Augment f along this path







Example: Round 2

Construct residual graph $G_f = (V, E_f)$

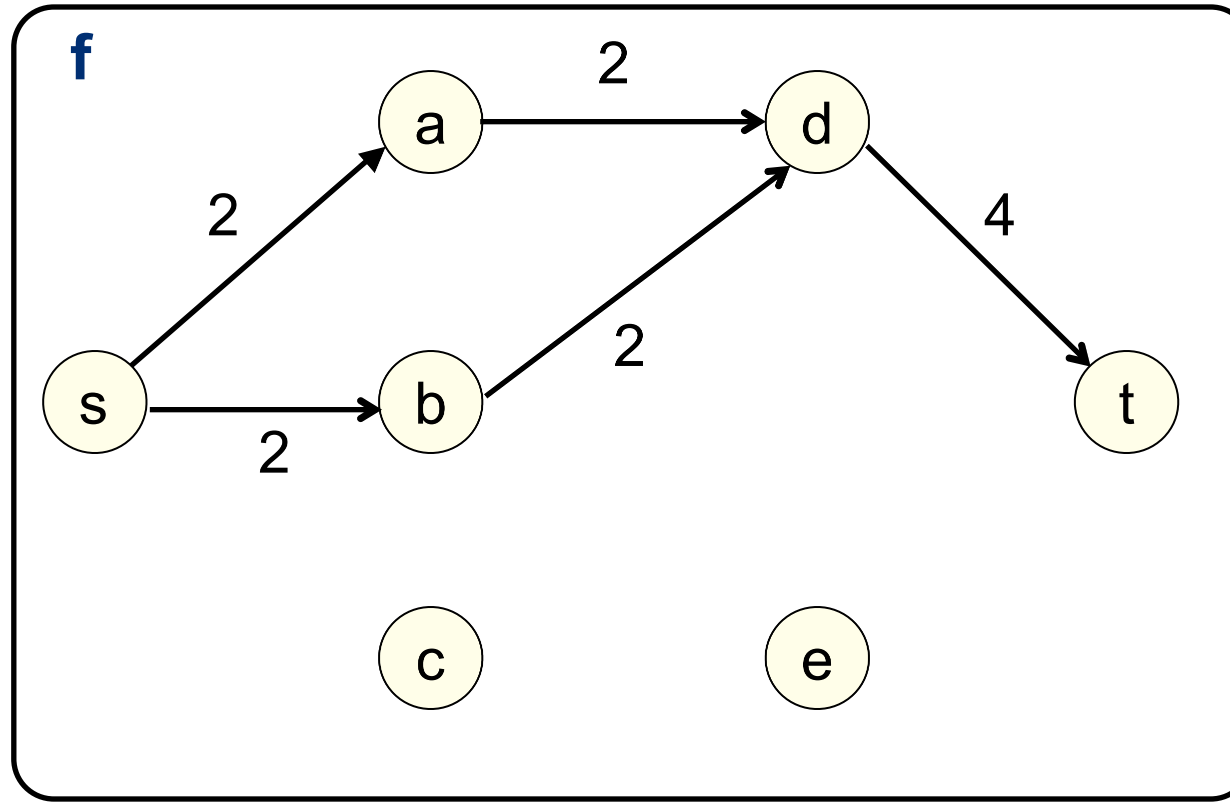
$E_f \subseteq E \cup E^R$

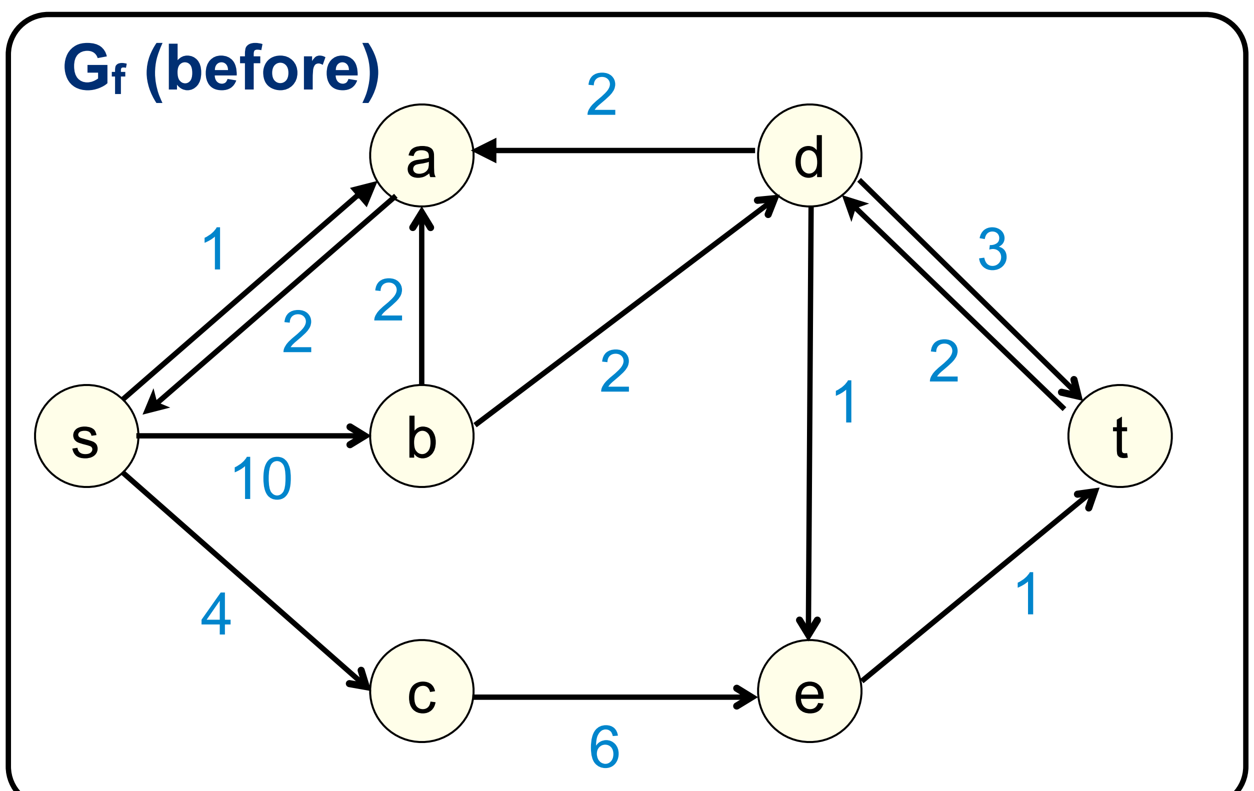
For any (u,v) in E or E^R,

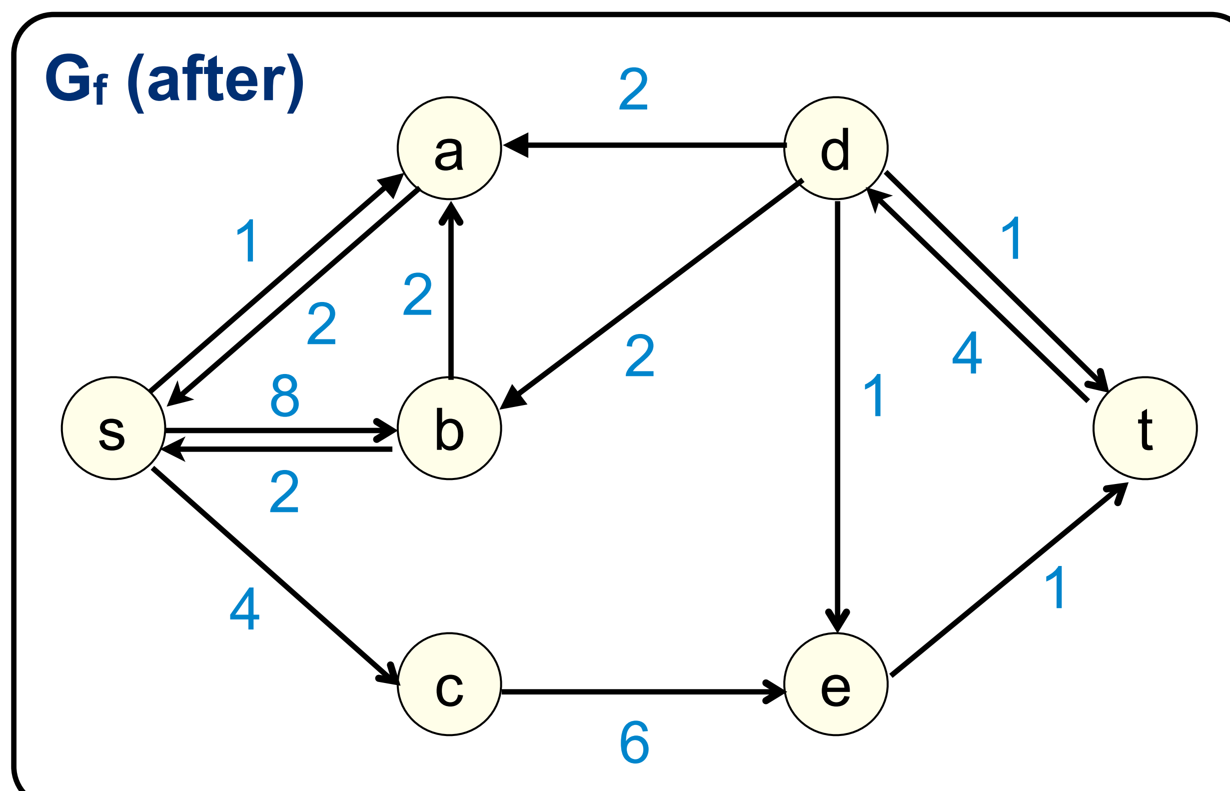
$c_f(u,v) = c(u,v) - f(u,v) + f(v,u)$

Find a path from s to t in G_f

Augment f along this path







Example: Round 3

Construct residual graph $G_f = (V, E_f)$

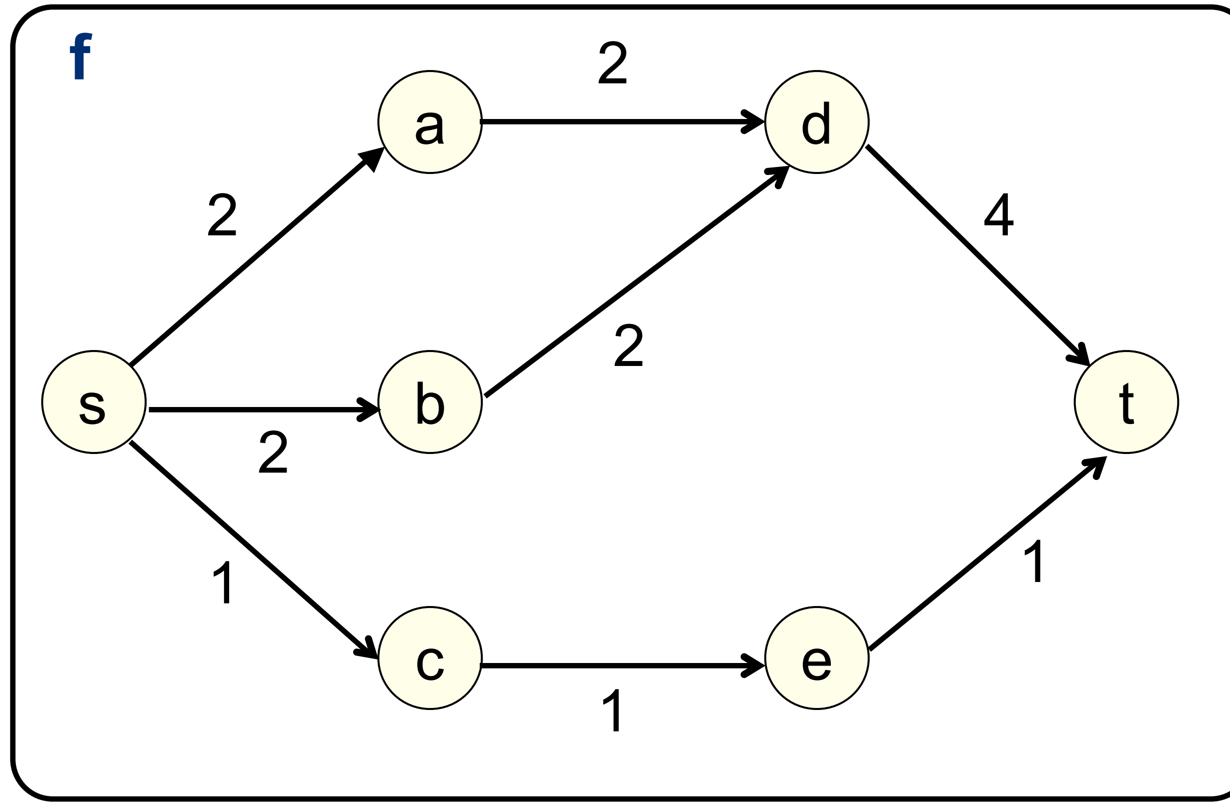
$E_f \subseteq E \cup E^R$

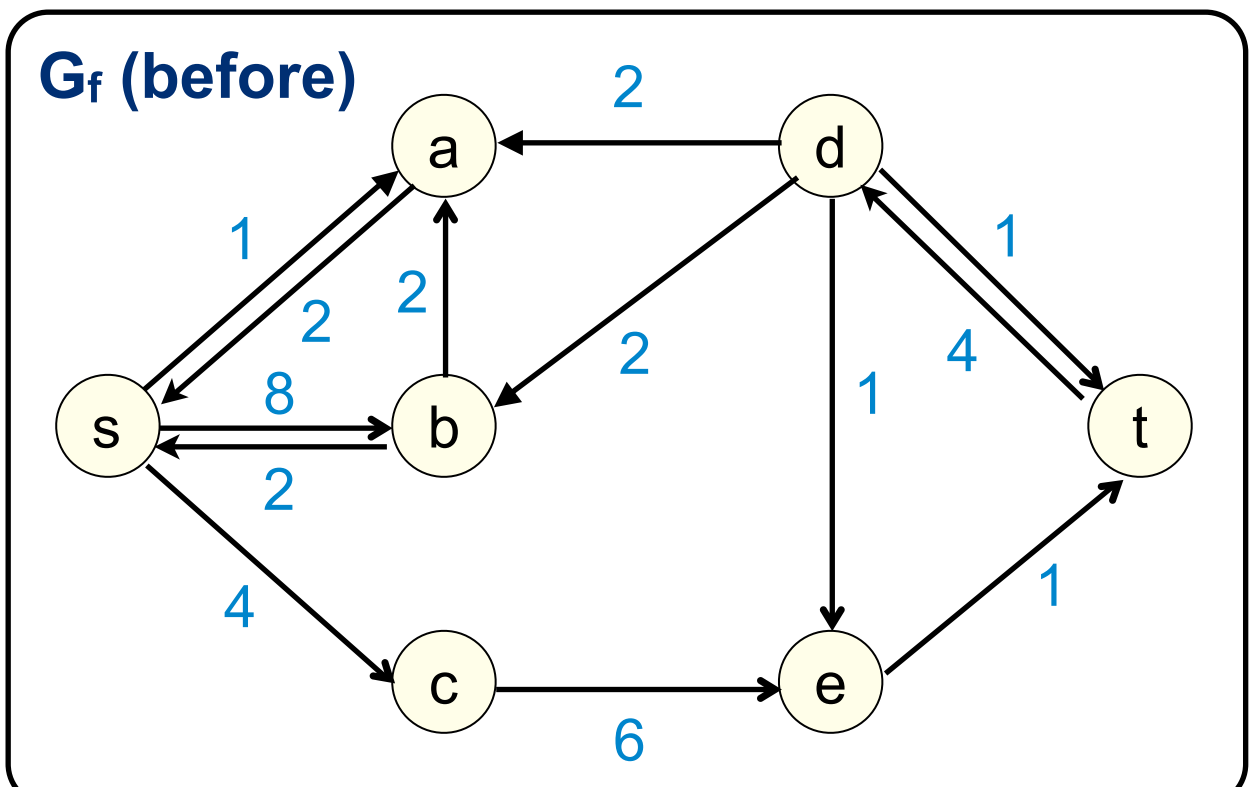
For any (u,v) in E or E^R,

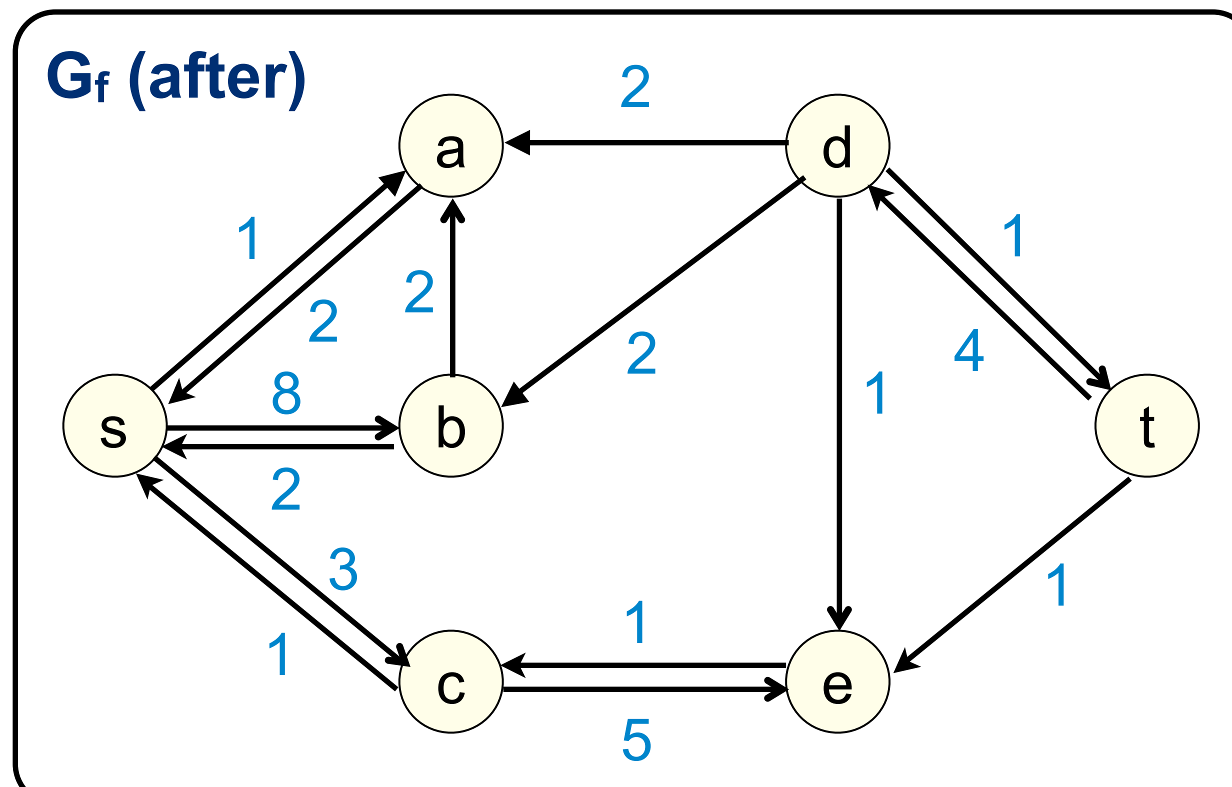
$c_f(u,v) = c(u,v) - f(u,v) + f(v,u)$

Find a path from s to t in G_f

Augment f along this path







Example: Round 3

Construct residual graph $G_f = (V, E_f)$

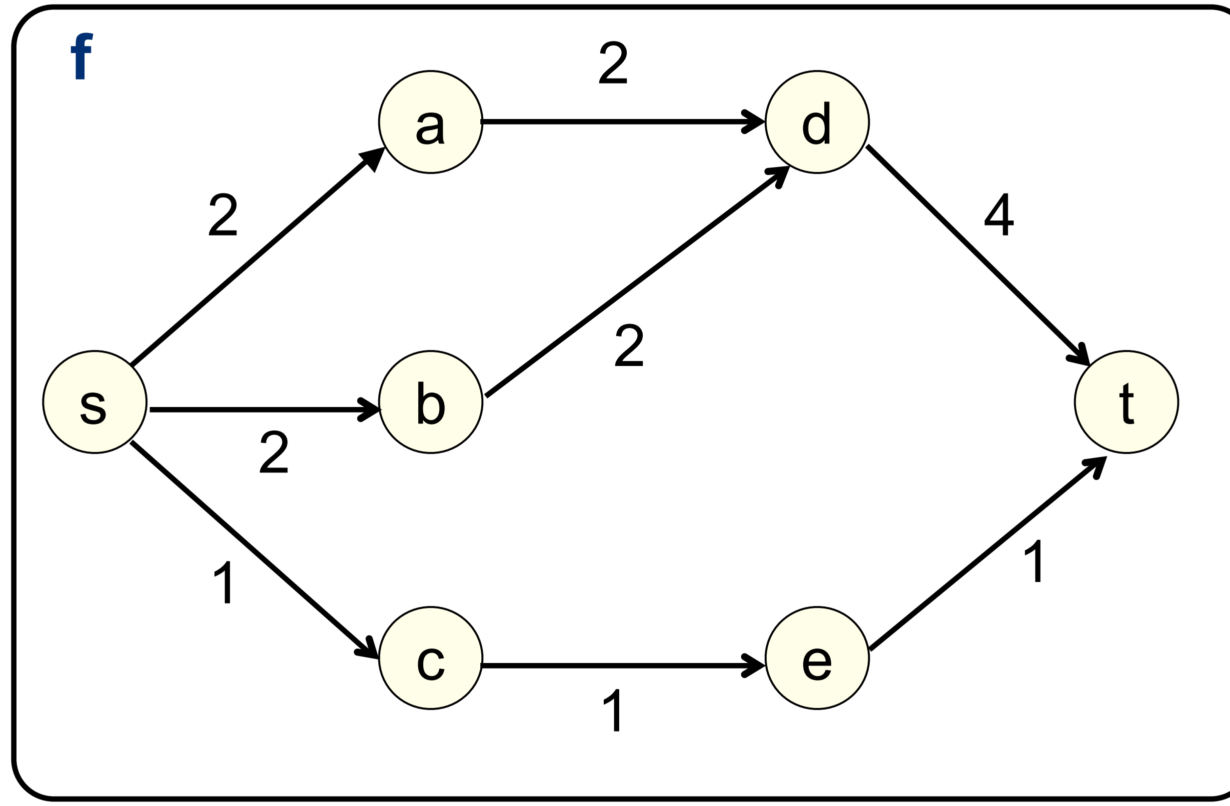
$E_f \subseteq E \cup E^R$

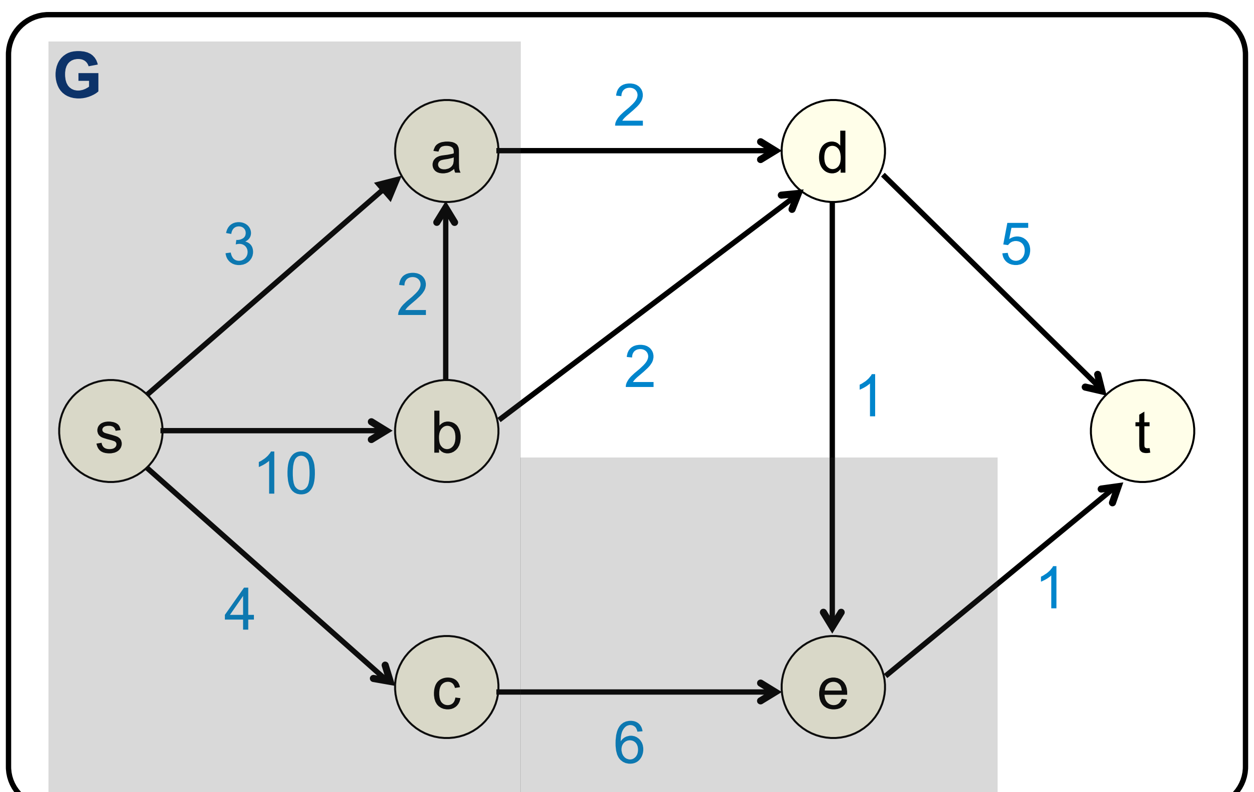
For any (u,v) in E or E^R,

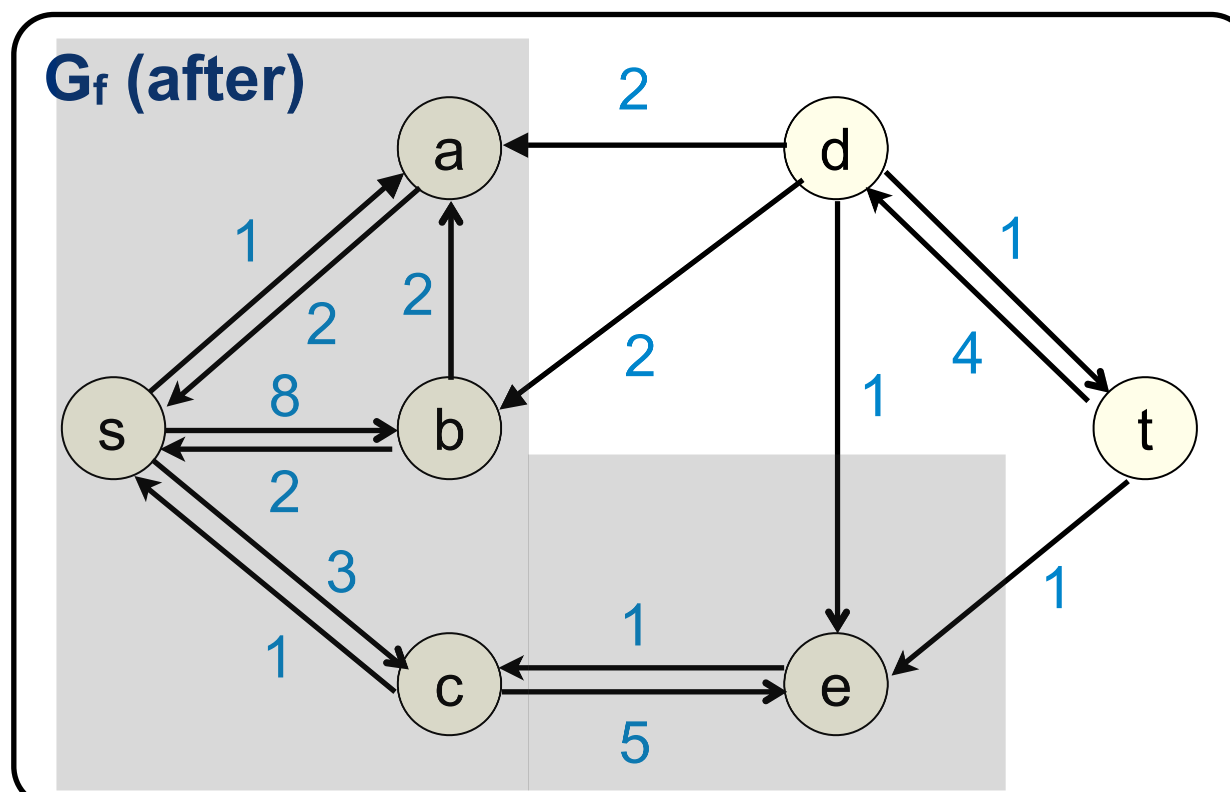
$c_f(u,v) = c(u,v) - f(u,v) + f(v,u)$

Find a path from s to t in G_f

Augment f along this path







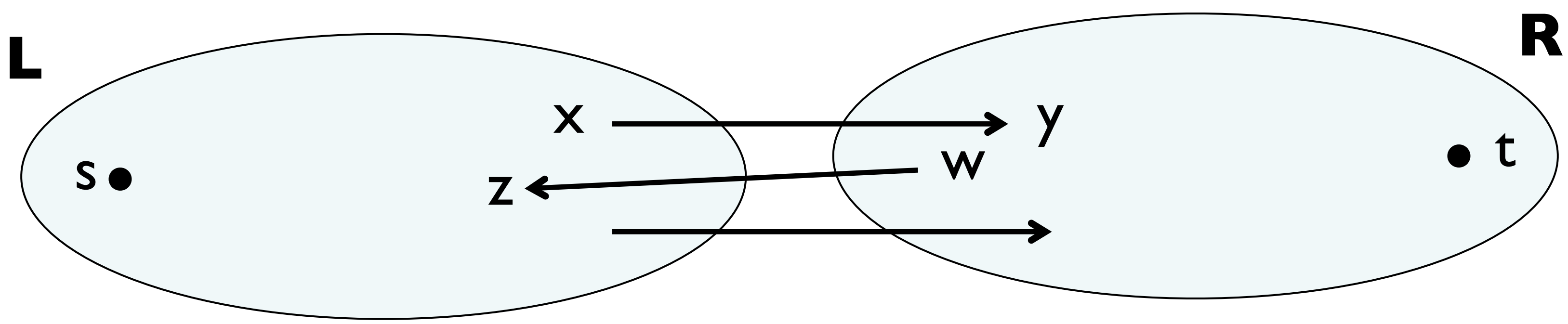
Analysis: Correctness

FF algorithm gives us a valid flow. But is it the **maximum** possible flow?

Consider final residual graph $G_f = (V, E_f)$

Let L = nodes reachable from s in G_f and let R = rest of nodes = V – L

So $s \in L$ and $t \in R$



Edges from L to R must be at full capacity

Edges from R to L must be empty

Therefore, flow across cut (L,R) is

$$\sum_{(u,v) \in E, u \in L, v \in R} c(u,v)$$

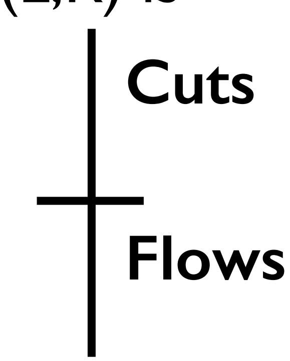
Thus, size(f) = capacity(L,R)

Recall: for any flow and any cut,

size(flow) <= capacity(cut)

Therefore f is the **max flow** and (L,R) is the **min cut**!

Thus, **Max Flow = Min Cut**



Analysis: efficiency

FF Algorithm: Start with zero flow

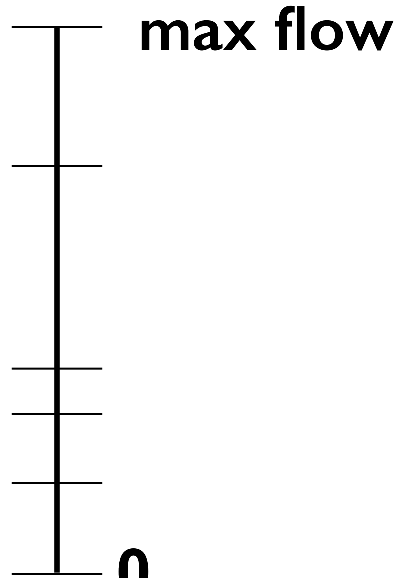
Repeat:

Find a path from s to t along which flow can be increased

Increase the flow along that path

A **hillclimbing** procedure

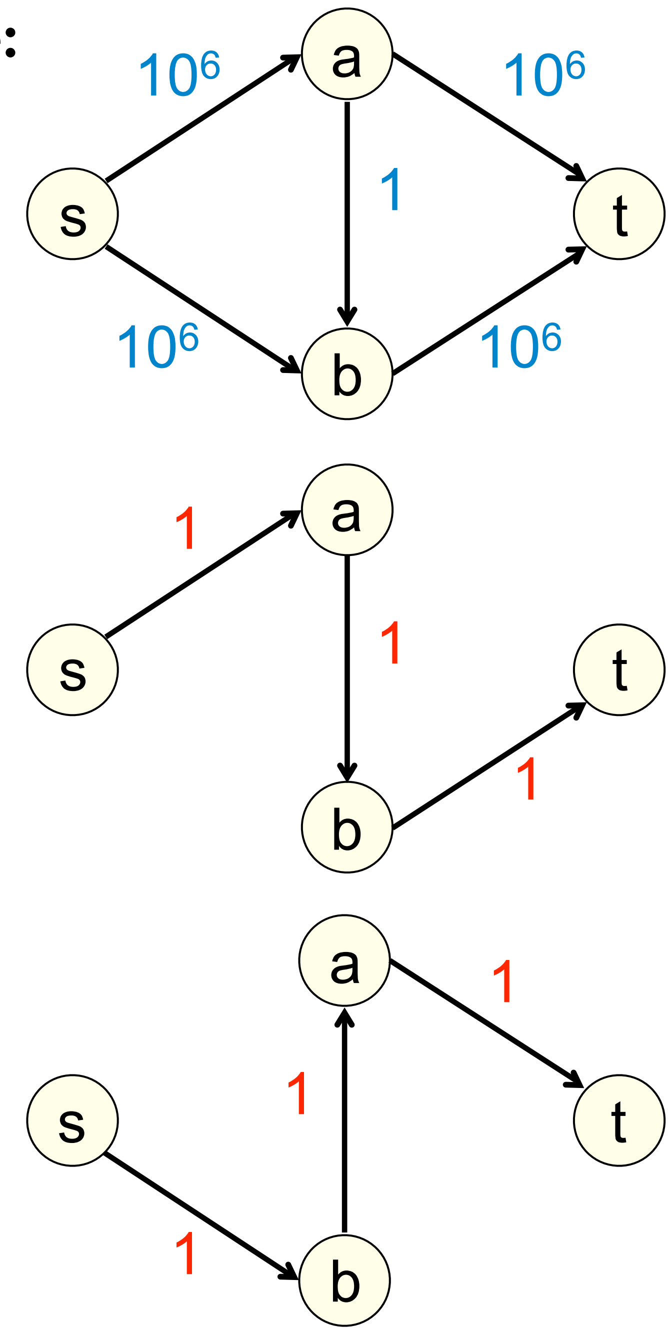
Flow size:



Each iteration is fast (O(|E|) time).

How many iterations are needed to reach the maximum flow?

Example:



#iterations can be Max Capacity

Analysis: efficiency

FF Algorithm: Start with zero flow

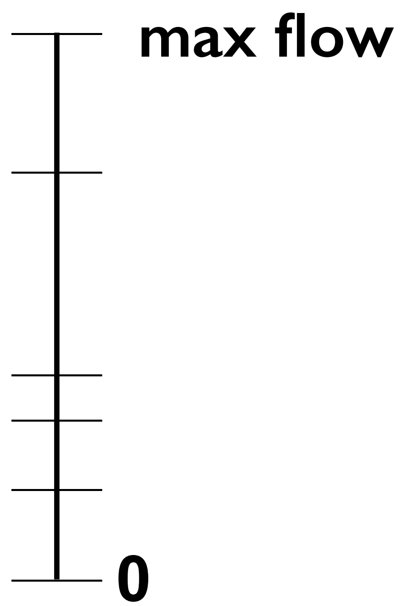
Repeat:

Find a path from s to t along which flow can be increased

Increase the flow along that path

A **hillclimbing** procedure

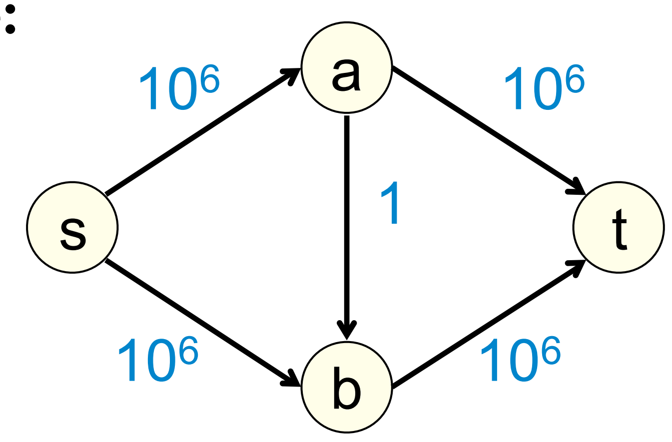
Flow size:



Each iteration is fast (O(|E|) time).

How many iterations are needed to reach the maximum flow?

Example:

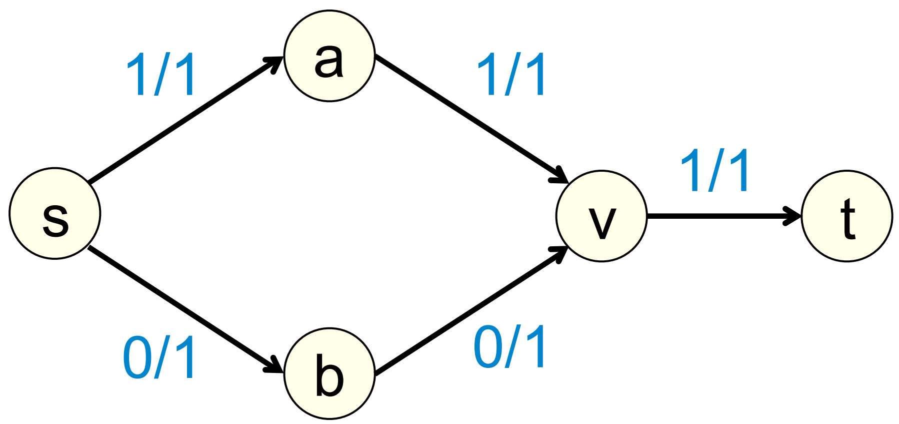


#iterations can be Max Capacity (with integer capacities)

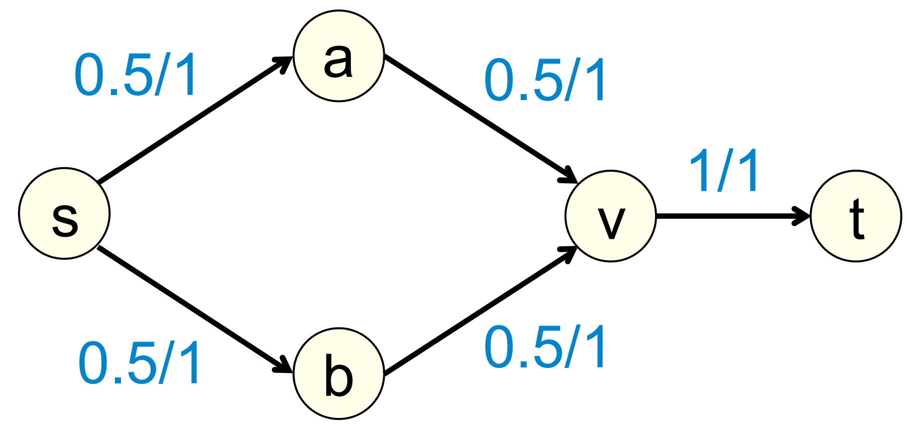
An Observation: Integrality

Integral Flows: A flow f is integral if f(e) is an integer for all e

Example:



An Integral Flow



A Fractional Flow

Property: If all edge capacities are integers, then, there is a max flow f which is integral.

Proof: If the edge capacities are integers, then, the FF algorithm always finds an integral flow
The FF algorithm also always finds a max flow.

Note: **All max flows** are not necessarily integral flows!

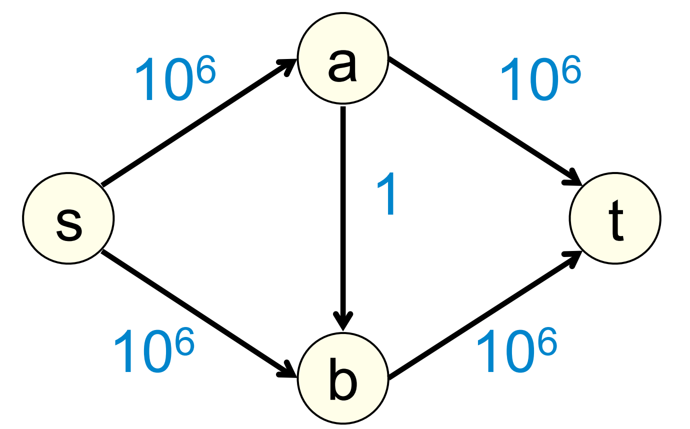
How to improve the efficiency?

- Ford-Fulkerson Style Algorithms:
  - Edmonds Karp
  - Capacity Scaling
- Preflow-Push

Edmonds Karp

FF Algorithm: Start with zero flow
Repeat:
  Find a path from s to t along which flow can be increased
  Increase the flow along that path

Bad Example for FF:



Bad Path Sequence:
(s, a, b, t), (s, b, a, t), (s, a, b, t),...

EK Path Selection: Find the **shortest path** along which flow can be increased (shortest path = shortest in terms of #edges)

It can be shown that this requires only O(|V||E|) iterations (Proof not in this class)
Running Time: $O(|V|\ |E|^2)$

Edmonds Karp

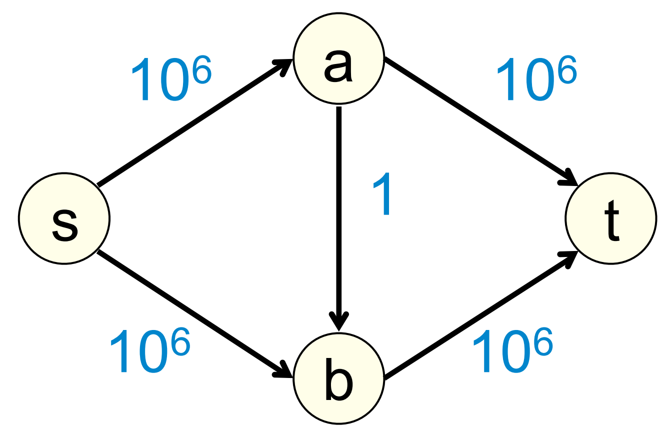
EK Algorithm: Start with zero flow

Repeat:

Find the **shortest path** from s to t along which flow can be increased

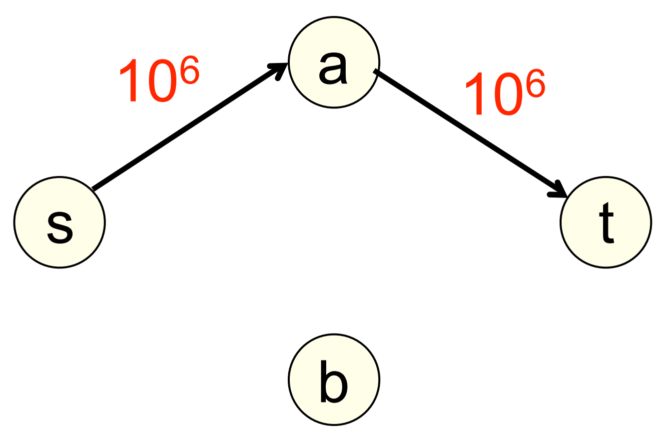
Increase the flow along that path

Bad Example for FF:

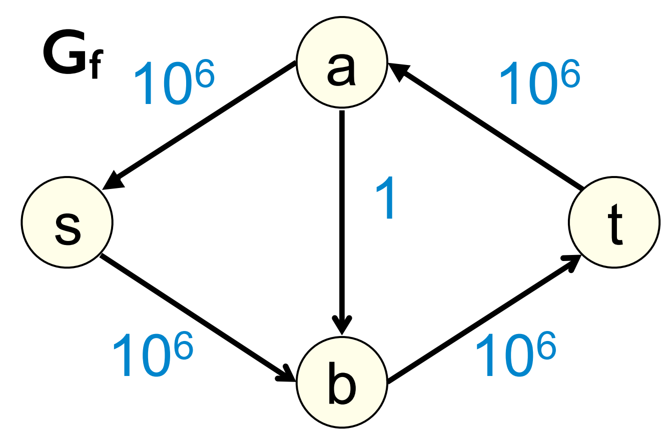


Iteration 1

f

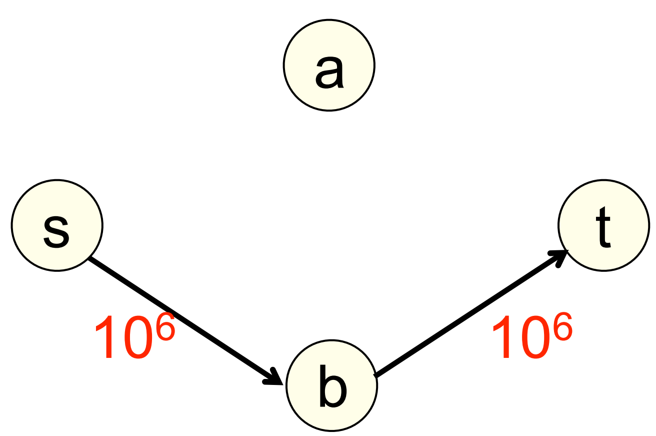


G_f

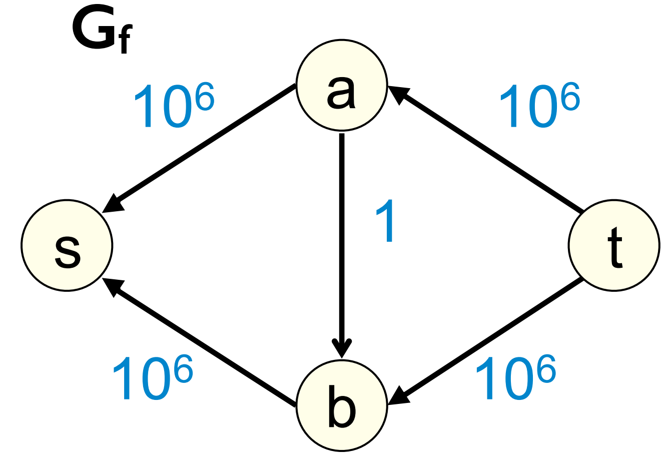


Iteration 2

f



G_f



How to improve the efficiency?

- Ford-Fulkerson Style Algorithms:
  - Edmonds Karp
  - Capacity Scaling

- Preflow-Push

Capacity Scaling

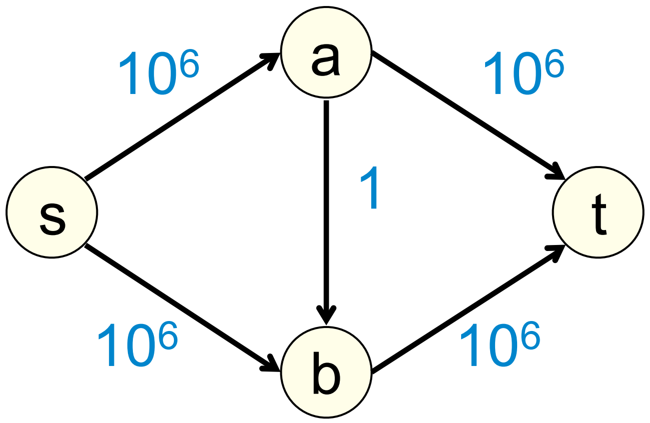
FF Algorithm: Start with zero flow

Repeat:

Find a path from s to t along which flow can be increased

Increase the flow along that path

Bad Example:



Bad Path Sequence:

(s, a, b, t), (s, b, a, t), (s, a, b, t),...

Capacity Scaling: Find **paths of high capacity first** between s and t

Capacity Scaling

C_{max} = max capacity edge. Start with D = C_{max}

Start with zero flow

While D >= 1, repeat:

  $G_f(D)$ = D-residual graph

  While there is a path from s to t in $G_f(D)$ along which flow can be increased

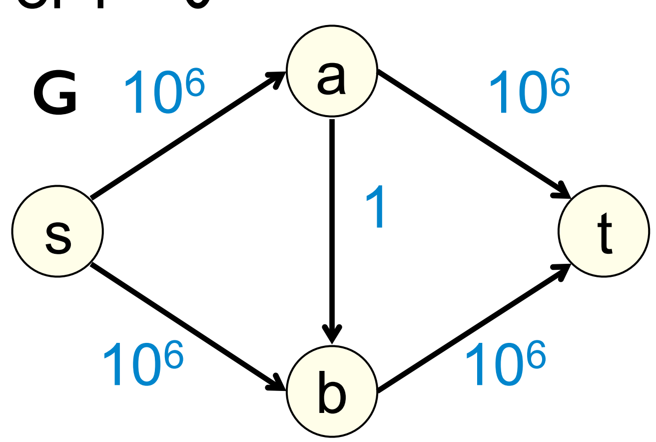
    Increase flow along that path

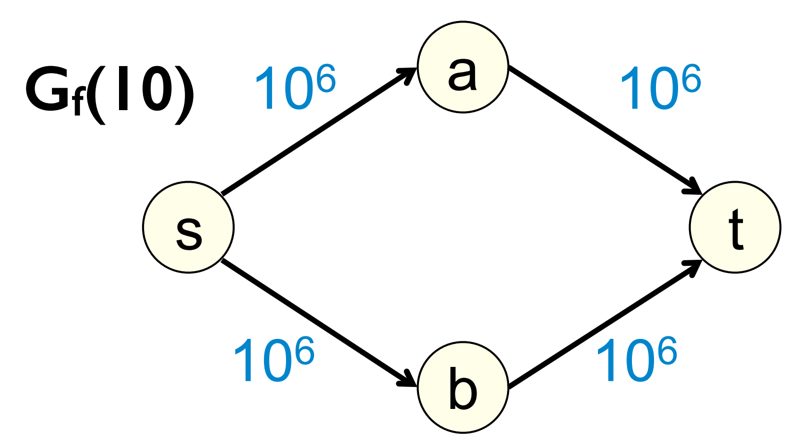
    Update $G_f(D)$

  D = D/2

D-Residual Graph: Subgraph of residual graph with only edges with capacity >= D

Example: For f = 0





Capacity Scaling: Correctness

C_{max} = max capacity edge. Start with D = C_{max}
Start with zero flow
While D >= 1, repeat:
    $G_f(D)$ = D-residual graph
    While there is a path from s to t in $G_f(D)$ along which flow can be increased
        Increase flow along that path
        Update $G_f(D)$
    D = D/2

D-Residual Graph: Subgraph of residual graph with only edges with capacity >= D

Property: If all edge capacities are integers, algorithm outputs a max flow

Proof: At D=1, $G_f(D) = G_f$. So on termination, $G_f(D)$ has no more paths from s to t

Capacity Scaling: Running Time

C_{max} = max capacity edge. Start with D = C_{max}

Start with zero flow

1 While D >= 1, repeat:

$G_f(D)$ = D-residual graph

D scaling phase

2 While there is a path from s to t in $G_f(D)$ along which flow can be increased

Increase flow along that path

Update $G_f(D)$

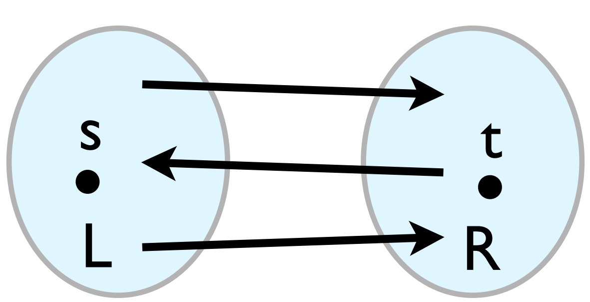
D = D/2

D-Residual Graph: Subgraph of residual graph with only edges with capacity >= D

Property 1: While loop 1 is executed 1 + $\log_2 C_{max}$ times

Property 2: At the end of a D-scaling phase, size(max flow) <= size(current flow) + D|E|

Proof: Let L = nodes reachable from s in $G_f(D)$ and let R = rest of nodes = V – L

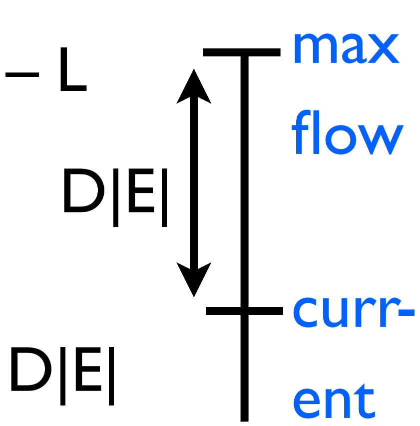


#edges in $G_f(D)$ in the (L, R) cut = 0

#edges in G_f in the (L,R) cut <= |E|

Capacity of each such edge < D

Thus, size(max flow) <= capacity(L,R) <= size(f) + D|E|



Capacity Scaling: Running Time

C_{max} = max capacity edge. Start with D = C_{max}

Start with zero flow

1 While D >= 1, repeat:

$G_f(D)$ = D-residual graph

D scaling phase

2 While there is a path from s to t in $G_f(D)$ along which flow can be increased

Increase flow along that path

Update $G_f(D)$

D = D/2

D-Residual Graph: Subgraph of residual graph with only edges with capacity >= D

Property 1: While loop 1 is executed $1 + \log_2 C_{max}$ times

Property 2: At the end of a D-scaling phase, size(max flow) <= size(current flow) + D|E|

Property 3: For any D, #iterations of loop 2 in the D-scaling phase <= 2|E|

Total Running Time: $O(|E|^2(1 + \log_2 C_{max}))$

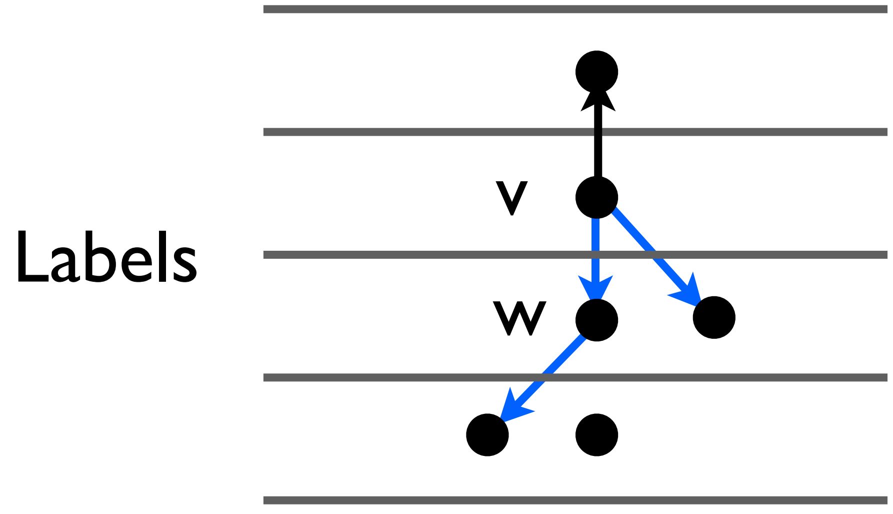
(Recall: Time to find a flow path in a residual graph = O(|E|))

How to improve the efficiency?

- Ford-Fulkerson Style Algorithms:
  - Edmonds Karp
  - Capacity Scaling
- Preflow-Push

Preflow-Push

Main Idea:
- Each node has a label, which is a potential
- Route flow from high to low potential



Idea: Route flow along blue edges

Preflows

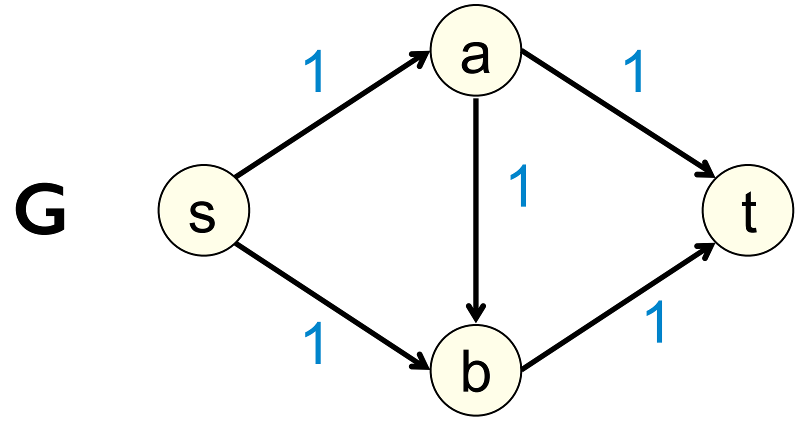
Preflow: A function f: E →R is a preflow if:

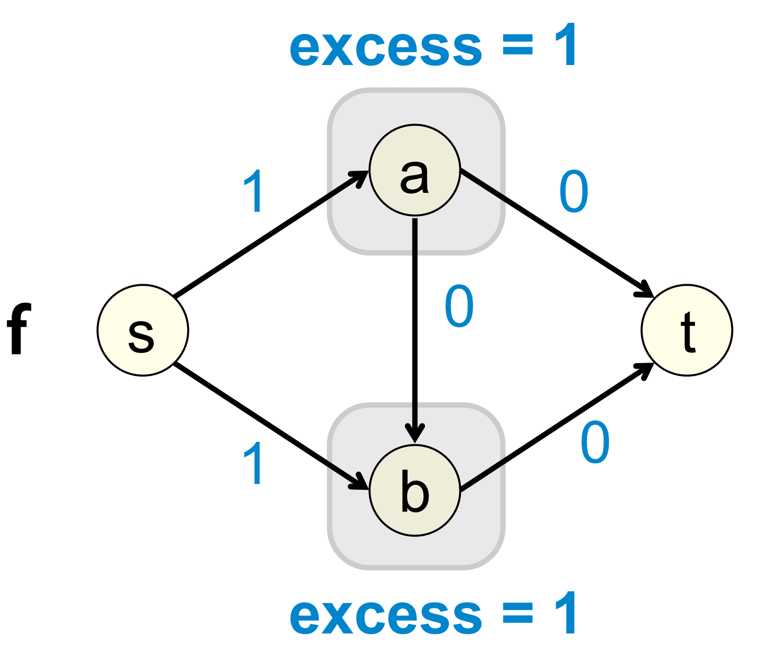
1. **Capacity Constraints:** 0 <= f(e) <= c(e)
2. Instead of conservation constraints:

$$\sum_{e \ into \ v} f(e) - \sum_{e \ out \ of \ v} f(e) \geq 0$$

Excess(v) = $\displaystyle\sum_{e \ into \ v} f(e) - \sum_{e \ out \ of \ v} f(e)$

Example





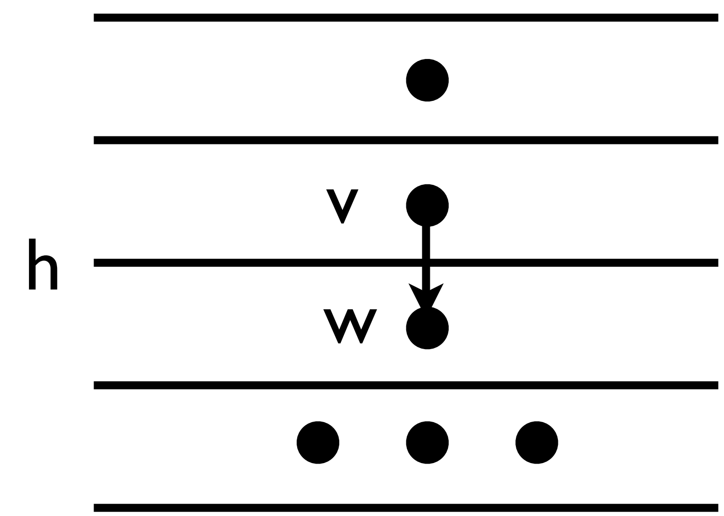
Preflow-Push: Two Operations

Preflow: A function f: E $\rightarrow$ R is a preflow if:
1. **Capacity Constraints:** 0 <= f(e) <= c(e)
2. Instead of conservation constraints:

$$\sum_{e\ into\ v} f(e) - \sum_{e\ out\ of\ v} f(e) \geq 0$$

Excess(v) = $\sum_{e\ into\ v} f(e) - \sum_{e\ out\ of\ v} f(e)$

Labeling h assigns a non-negative integer label h(v) to all v in V



Push(v, w): Applies if excess(v) > 0, h(w) < h(v), (v, w) in E_f
q = min(excess(v), c_f(v,w))
Add q to f(v, w)

Relabel(v): Applies if excess(v) > 0, for all w s.t (v, w) in E_f, h(w) >= h(v)
Increase h(v) by 1

Pre-Flow Push: The Algorithm

Start with labeling: h(s) = n, h(t) = 0, h(v) = 0, for all other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, for all other edges e

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f such that push(v, w) can be applied
    Push(v, w)
  Else
    Relabel(v)

Push(v, w): Applies if excess(v) > 0, h(w) < h(v), (v, w) in E_f
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v): Applies if excess(v) > 0, for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

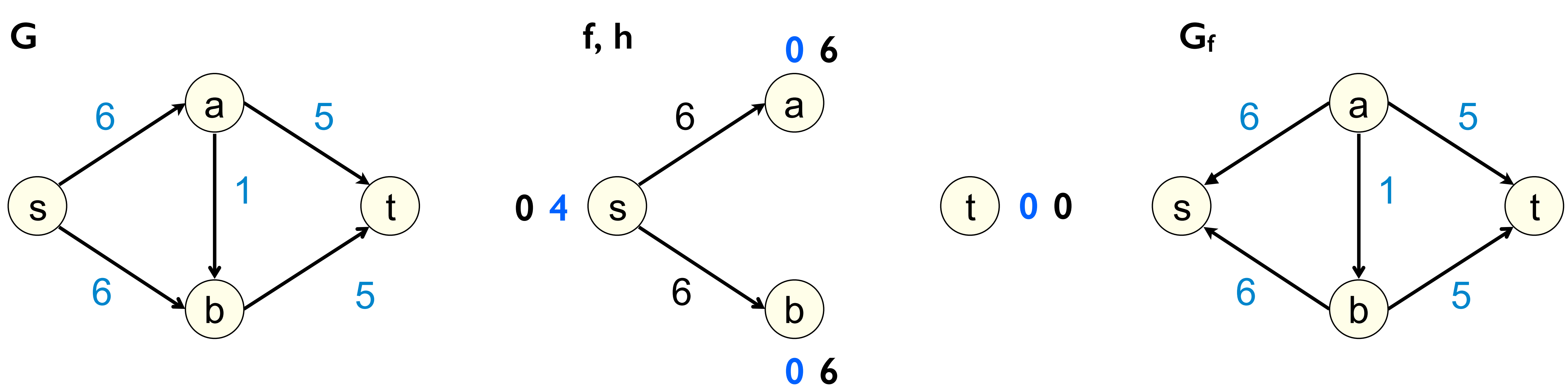
Pre-Flow Push: An Example

Start with labeling: h(s) = n, h(t) = 0, h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1



Labels
Excesses

Pre-Flow Push: An Example

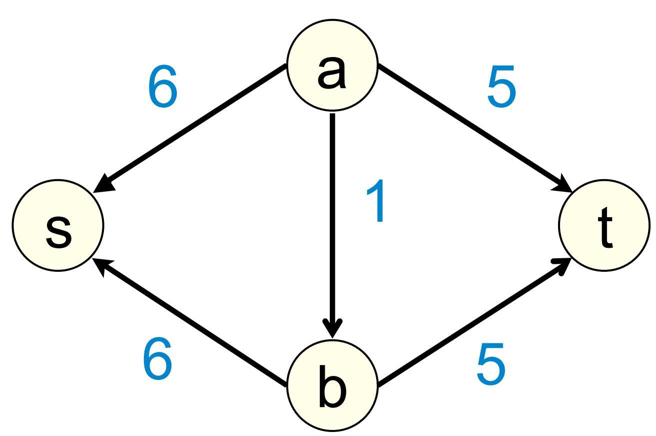
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

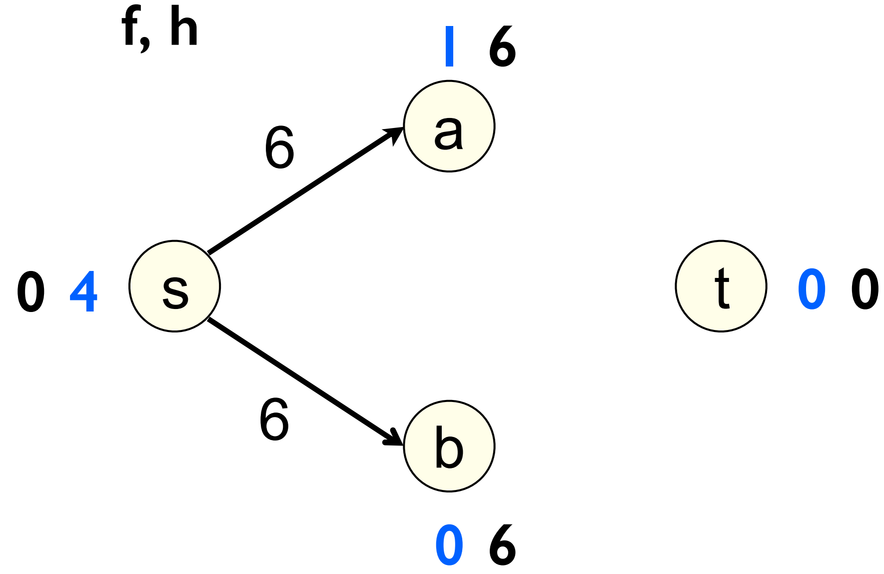
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    $q = \min(\text{excess}(v), c_f(v,w))$
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
    Increase h(v) by 1

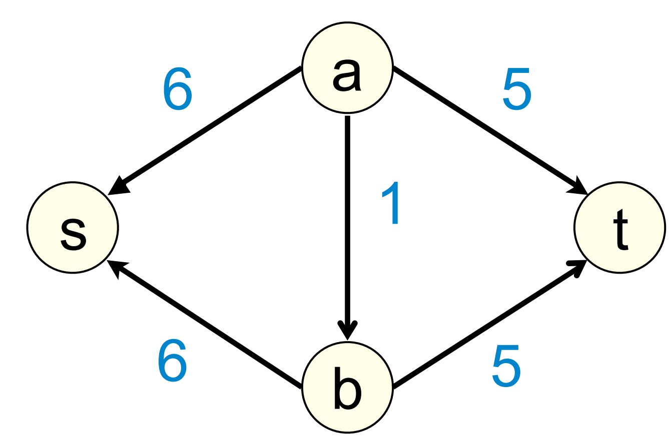
G_f (before)



f, h



G_f



Labels

Excesses

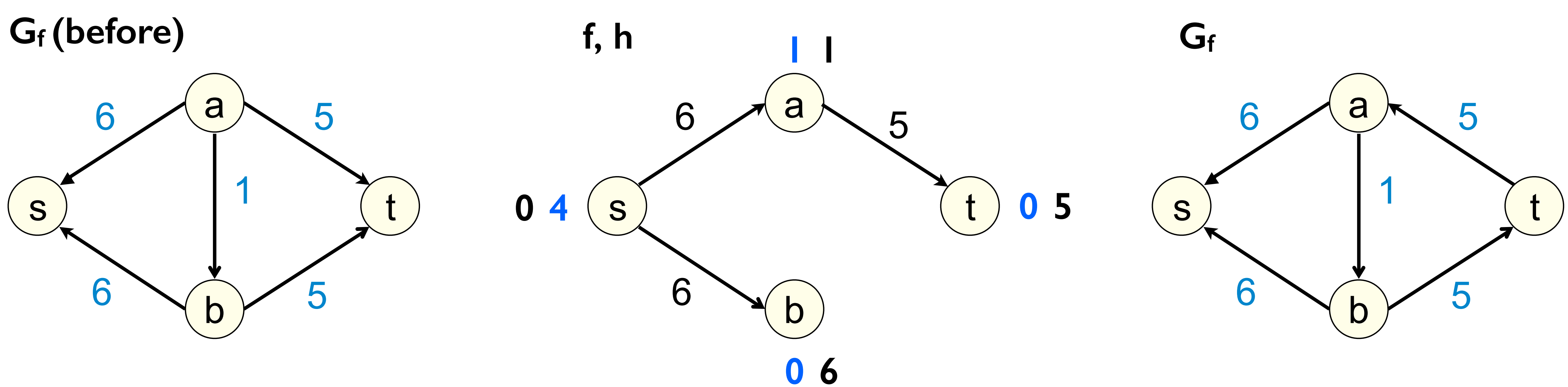
Pre-Flow Push: An Example

Start with labeling: h(s) = n, h(t) = 0, h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1



Labels
Excesses

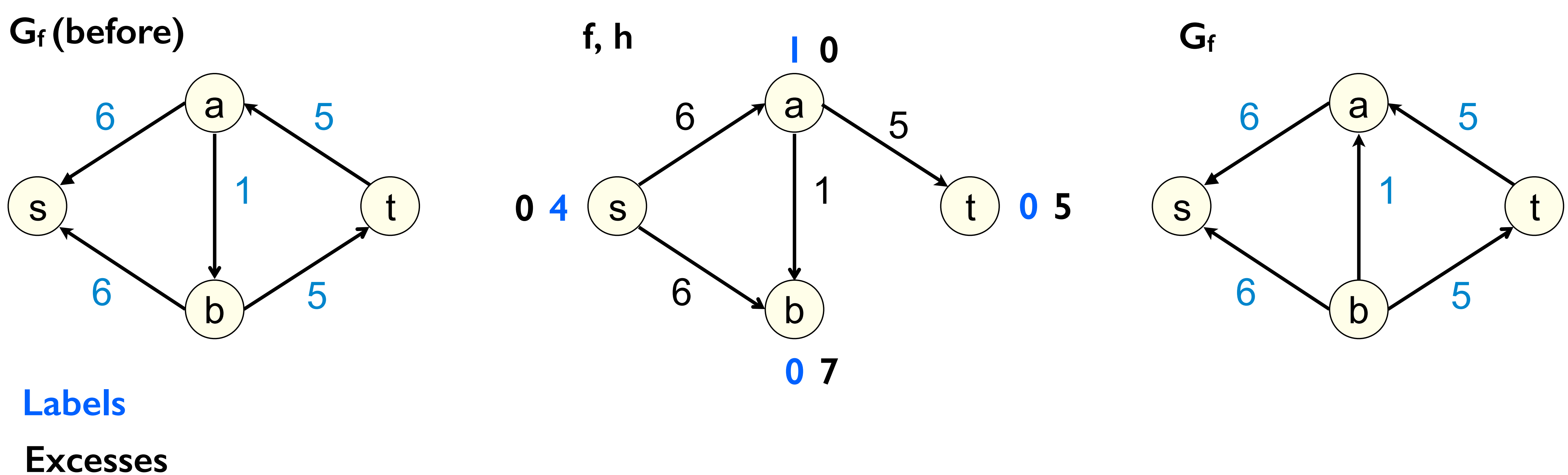
Pre-Flow Push: An Example

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  $q = \min(\text{excess}(v), c_f(v,w))$
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1



Pre-Flow Push: An Example

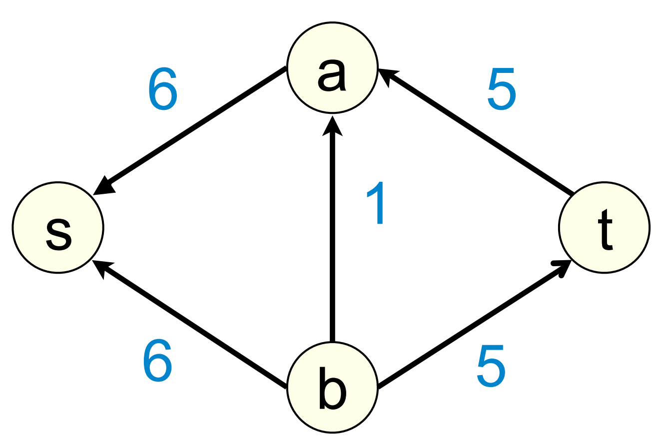
Start with labeling: h(s) = n, h(t) = 0, h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

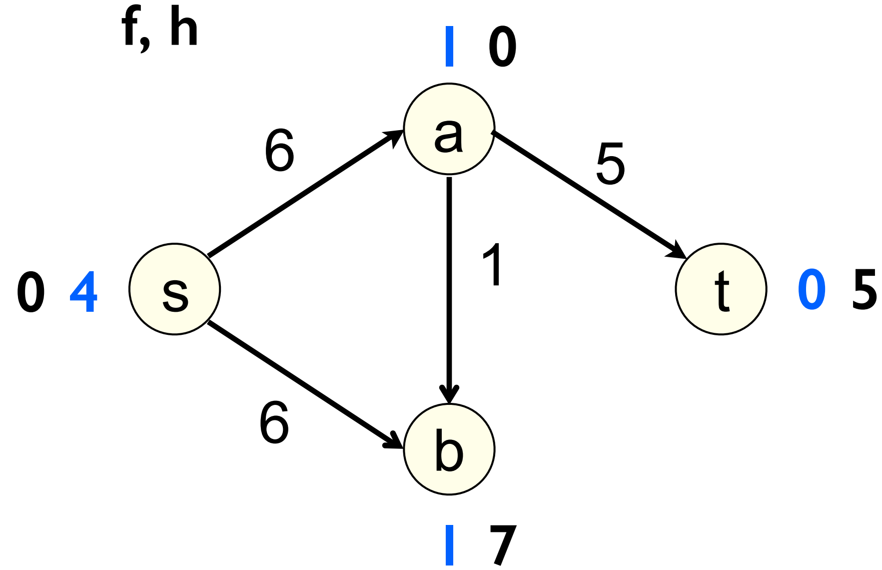
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

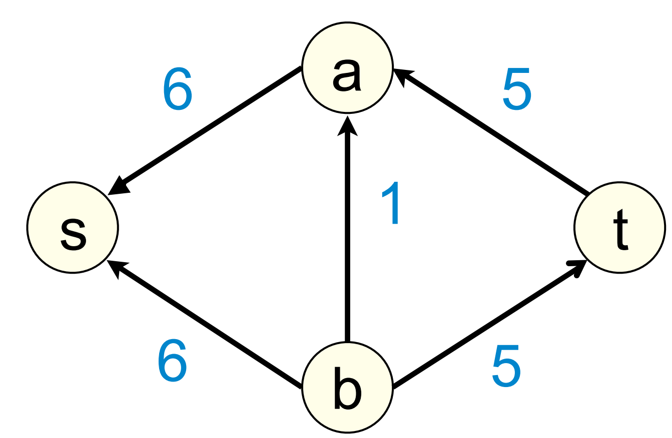
G_f (before)



f, h



G_f



Labels
Excesses

Pre-Flow Push: An Example

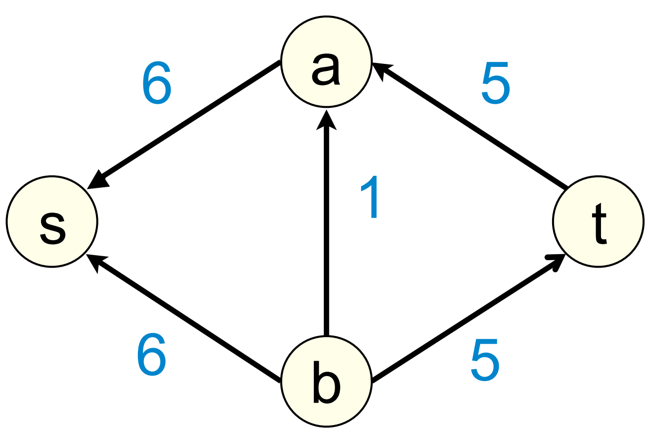
Start with labeling: $h(s) = n$, $h(t) = 0$, $h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

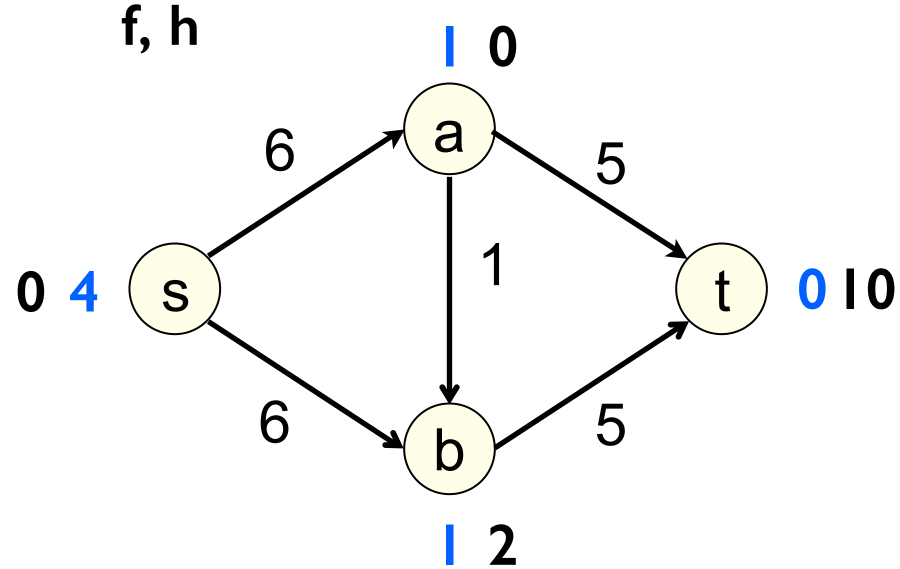
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    $q = \min(\text{excess}(v), c_f(v,w))$
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

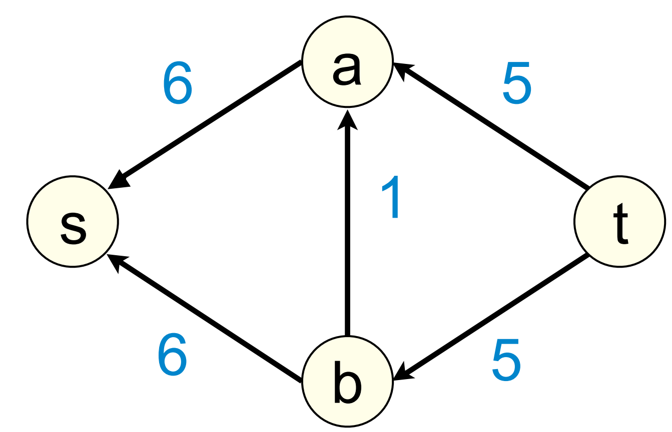
G_f (before)



f, h



G_f



Labels
Excesses

Pre-Flow Push: An Example

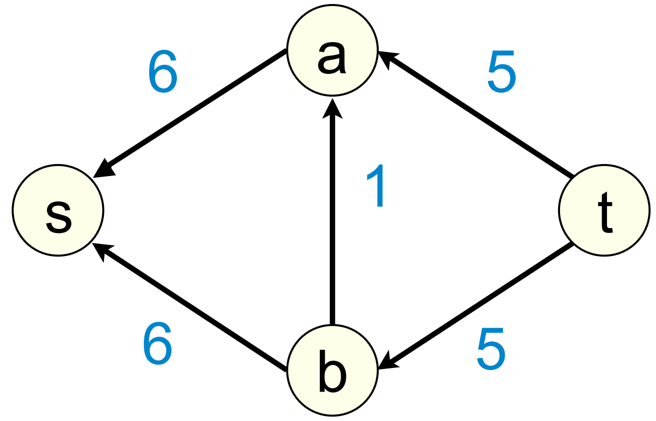
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
- Pick a node v with excess(v) > 0
- If there is an edge (v, w) in E_f s. t. push(v, w) applies
  - Push(v, w)
- Else
  - Relabel(v)

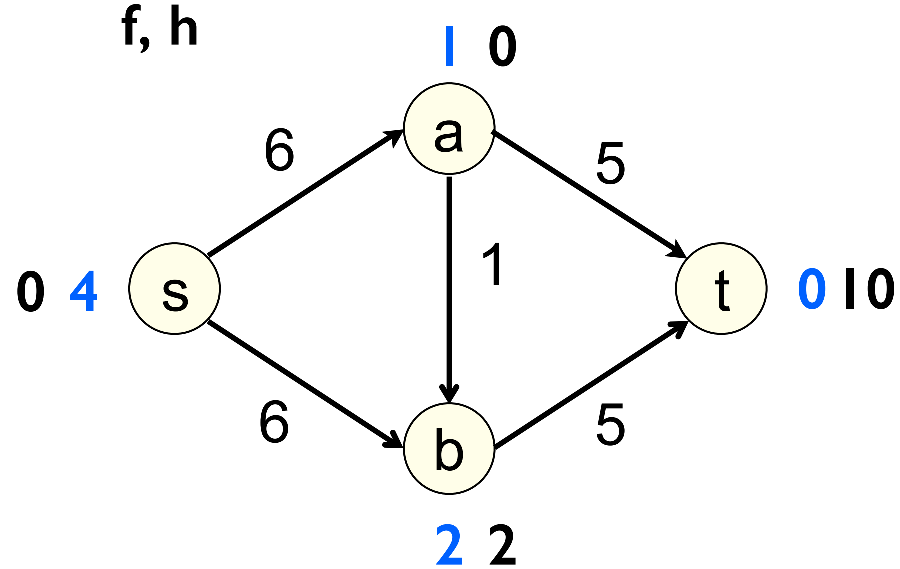
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
- $q = \min(\text{excess}(v), c_f(v,w))$
- Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
- Increase h(v) by 1

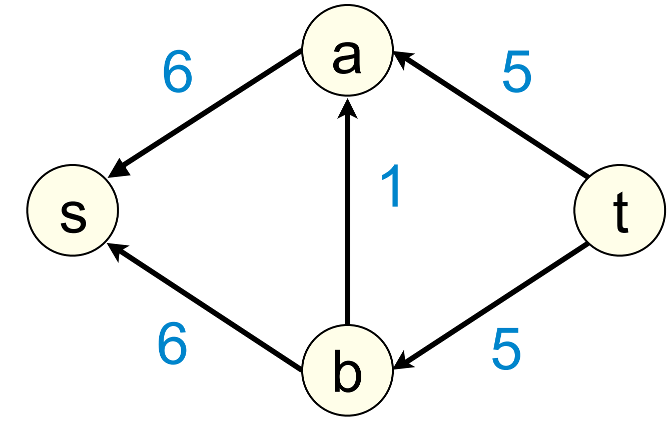
G_f (before)



f, h



G_f



Labels

Excesses

Pre-Flow Push: An Example

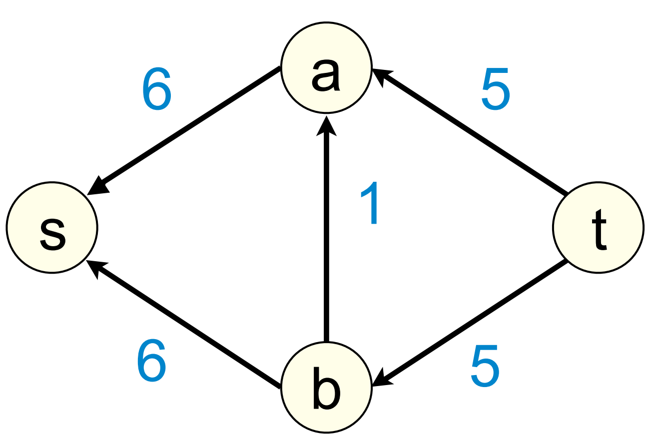
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

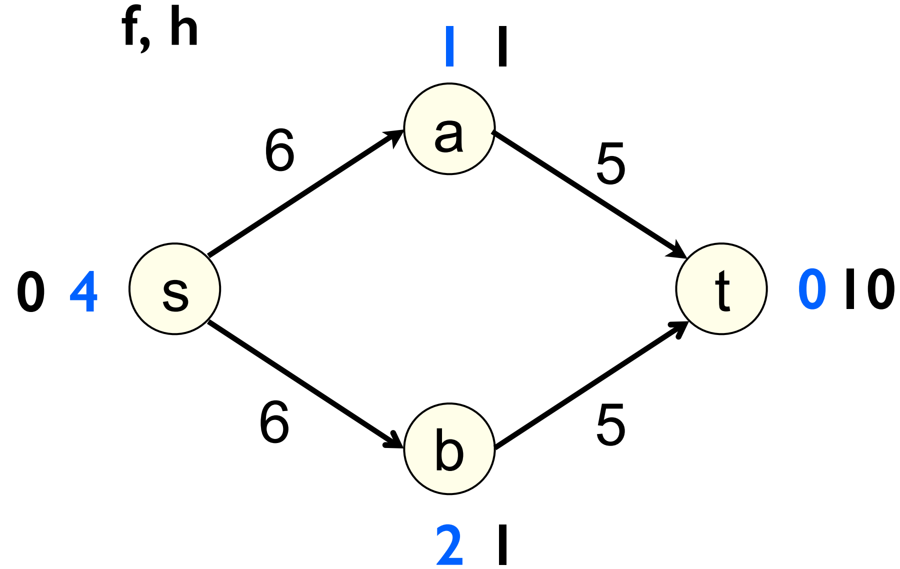
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  $q = \min(\text{excess}(v), c_f(v,w))$
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

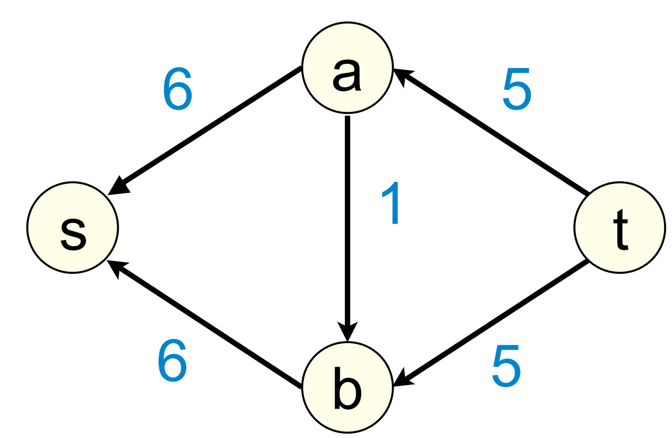
G_f (before)



f, h



G_f



Labels
Excesses

Pre-Flow Push: An Example

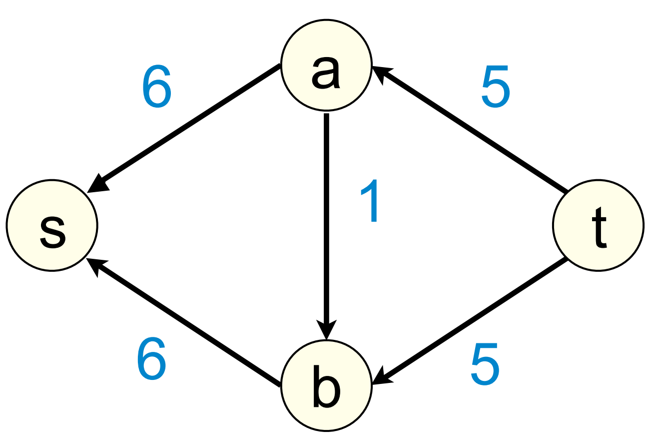
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

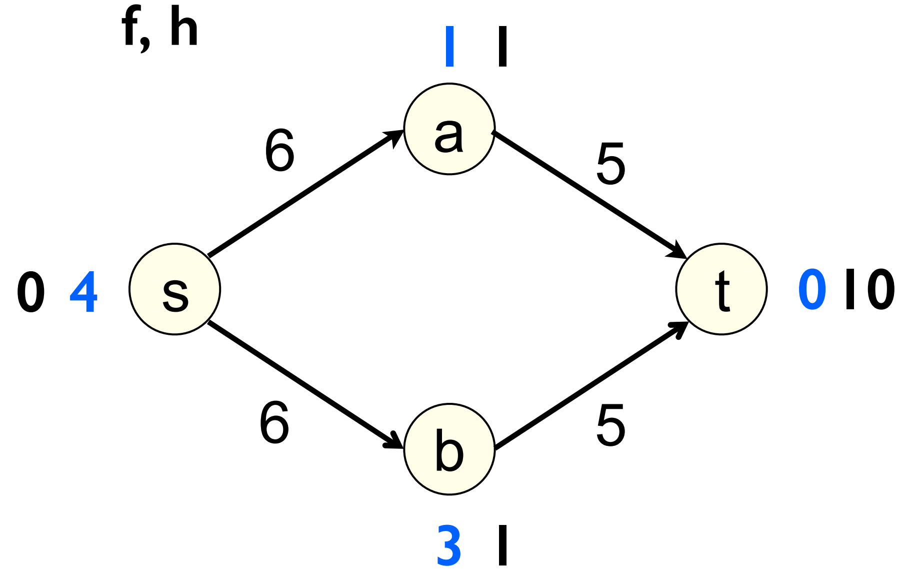
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

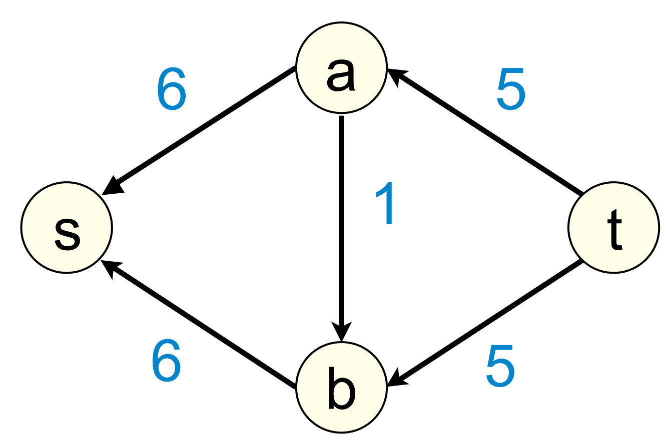
G_f (before)



f, h



G_f



Labels

Excesses

Pre-Flow Push: An Example

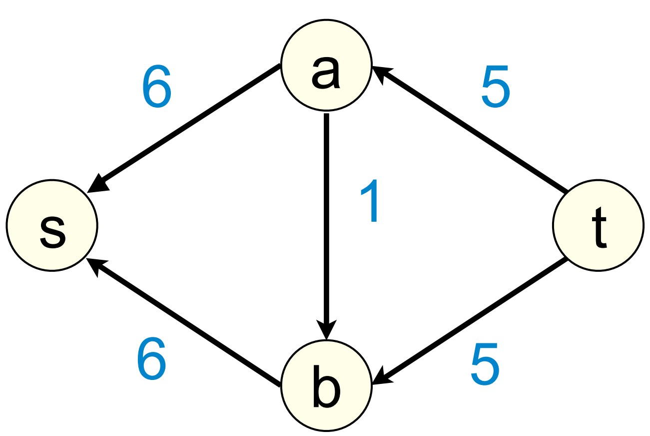
Start with labeling: h(s) = n, h(t) = 0, h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

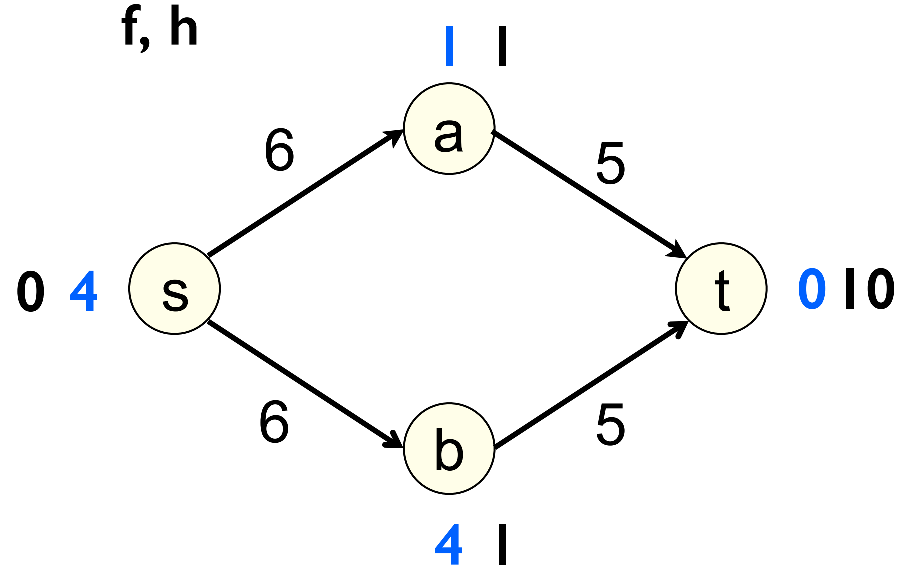
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

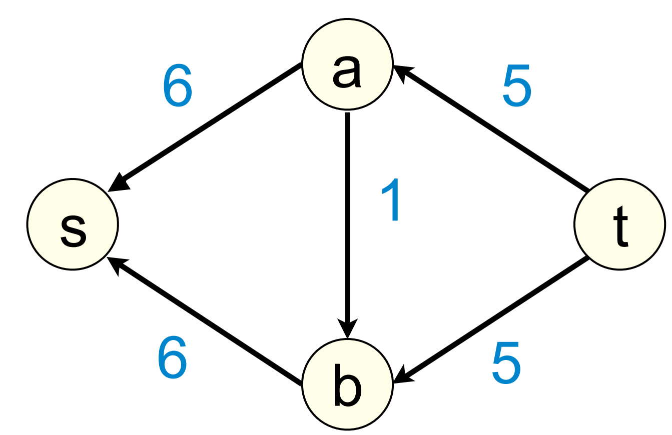
G_f (before)



f, h



G_f



Labels
Excesses

Pre-Flow Push: An Example

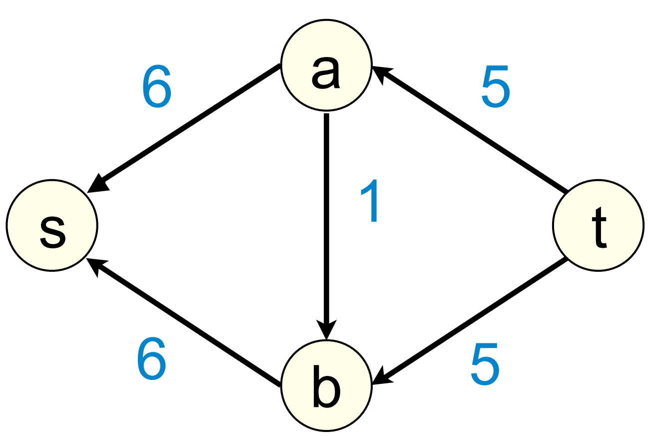
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

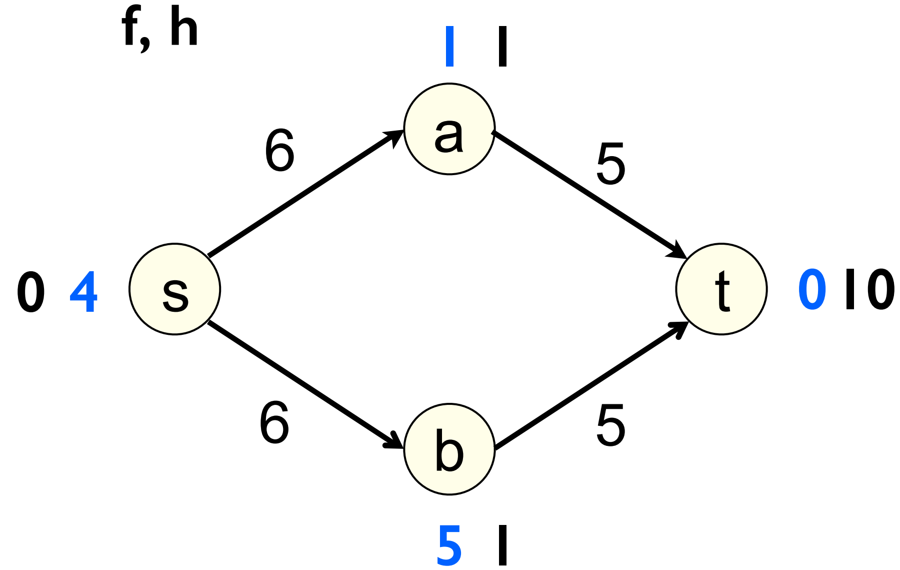
Push(v, w):
Applies if excess(v) > 0, $h(w) < h(v)$
  q = min(excess(v), $c_f(v,w)$)
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
  Increase h(v) by 1

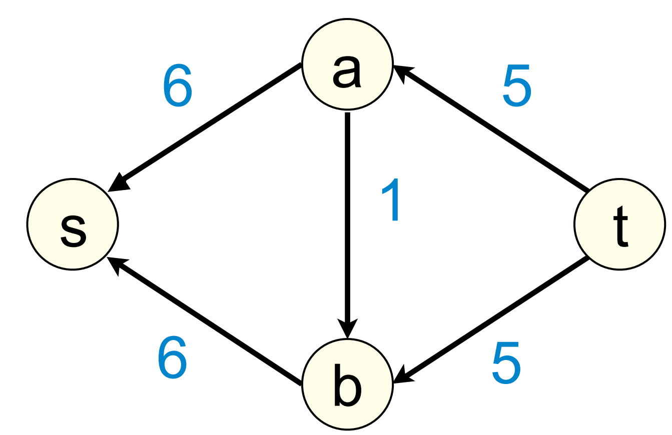
G_f (before)



f, h



G_f



Labels

Excesses

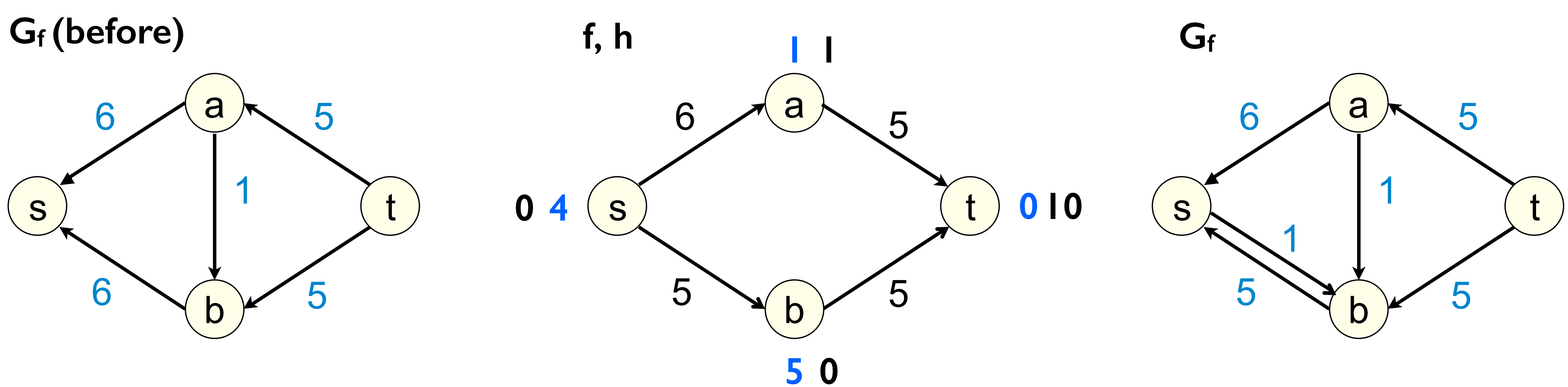
Pre-Flow Push: An Example

Start with labeling: $h(s) = n$, $h(t) = 0$, $h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
- Pick a node v with excess(v) > 0
- If there is an edge (v, w) in E_f s. t. push(v, w) applies
  - Push(v, w)
- Else
  - Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
- $q = \min(\text{excess}(v), c_f(v,w))$
- Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
- Increase h(v) by 1



Labels

Excesses

Pre-Flow Push: An Example

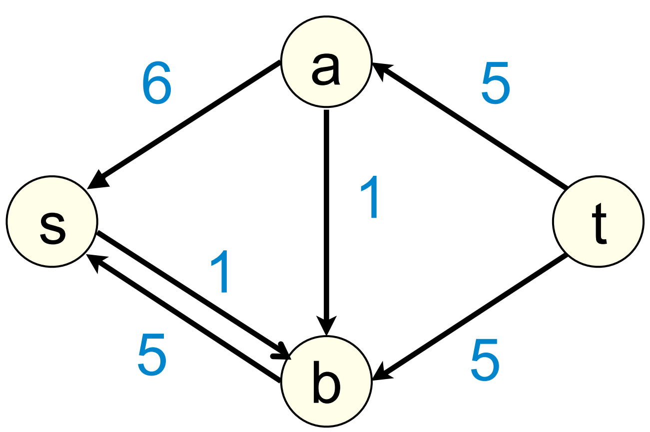
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

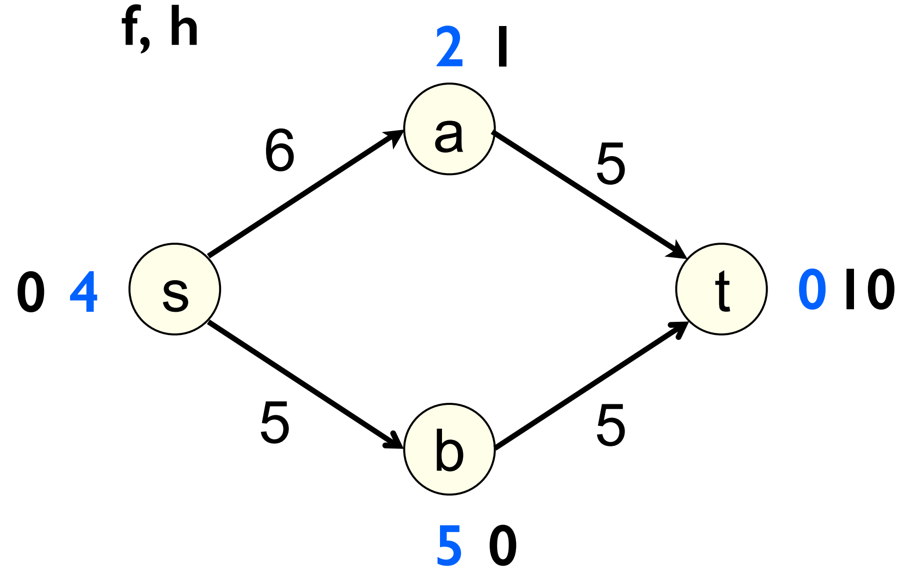
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

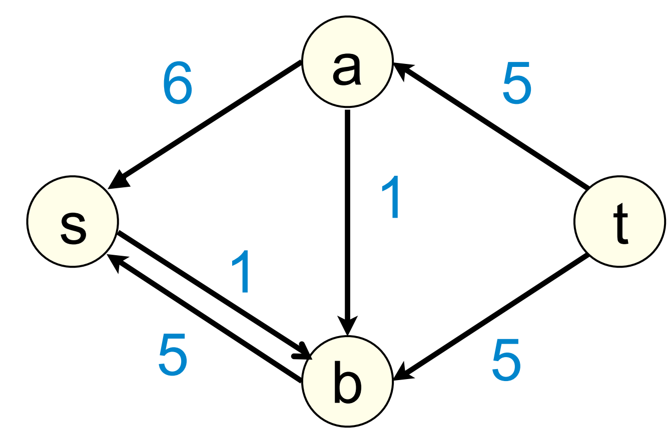
G_f (before)



f, h



G_f



Labels

Excesses

Pre-Flow Push: An Example

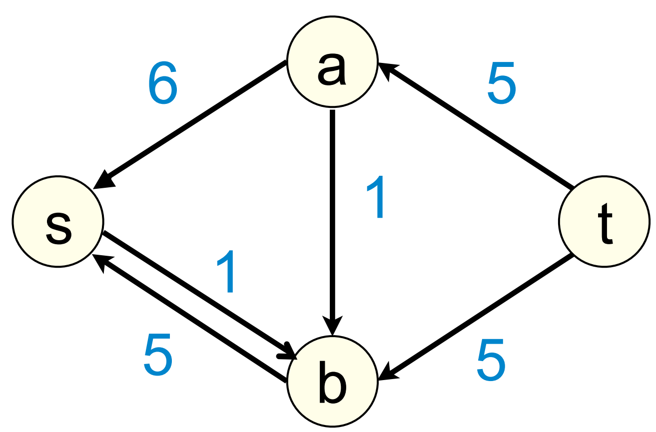
Start with labeling: h(s) = n, h(t) = 0, h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

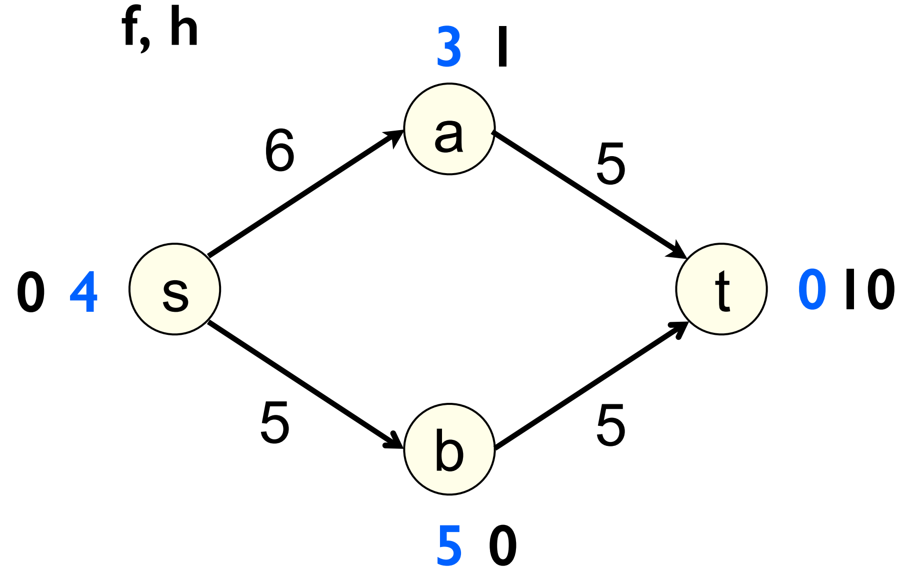
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

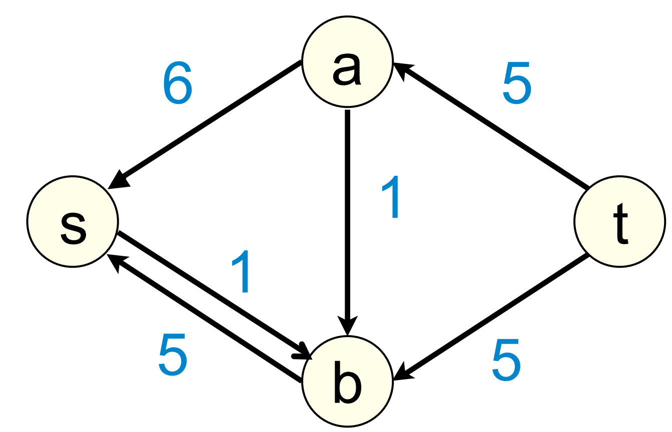
G_f (before)



f, h



G_f



Labels

Excesses

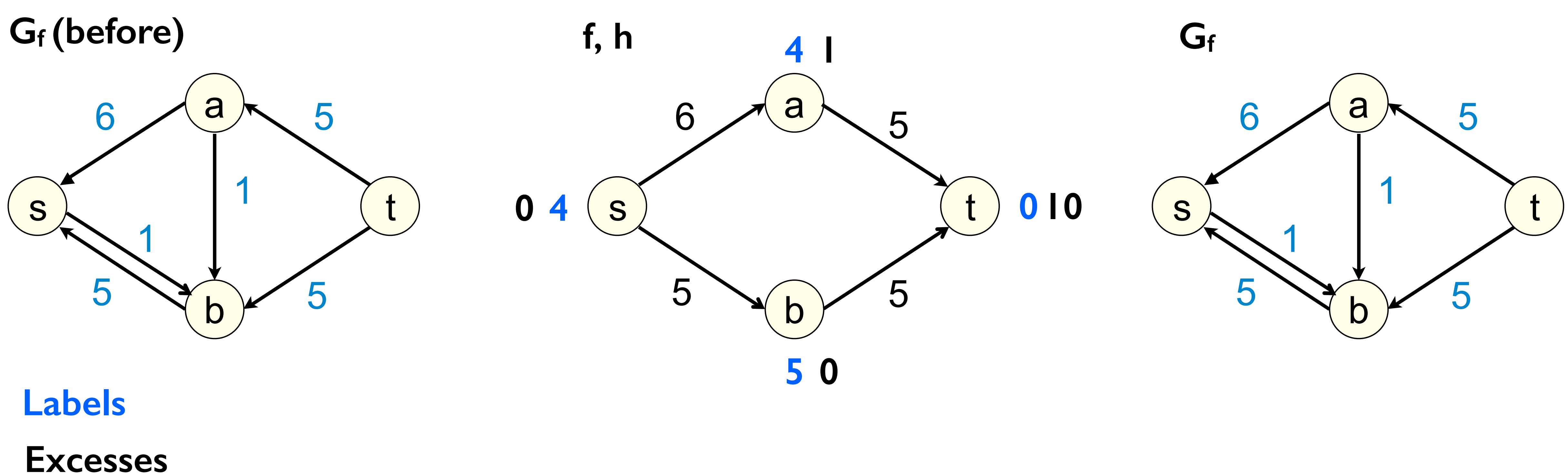
Pre-Flow Push: An Example

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  $q = \min(\text{excess}(v), c_f(v,w))$
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1



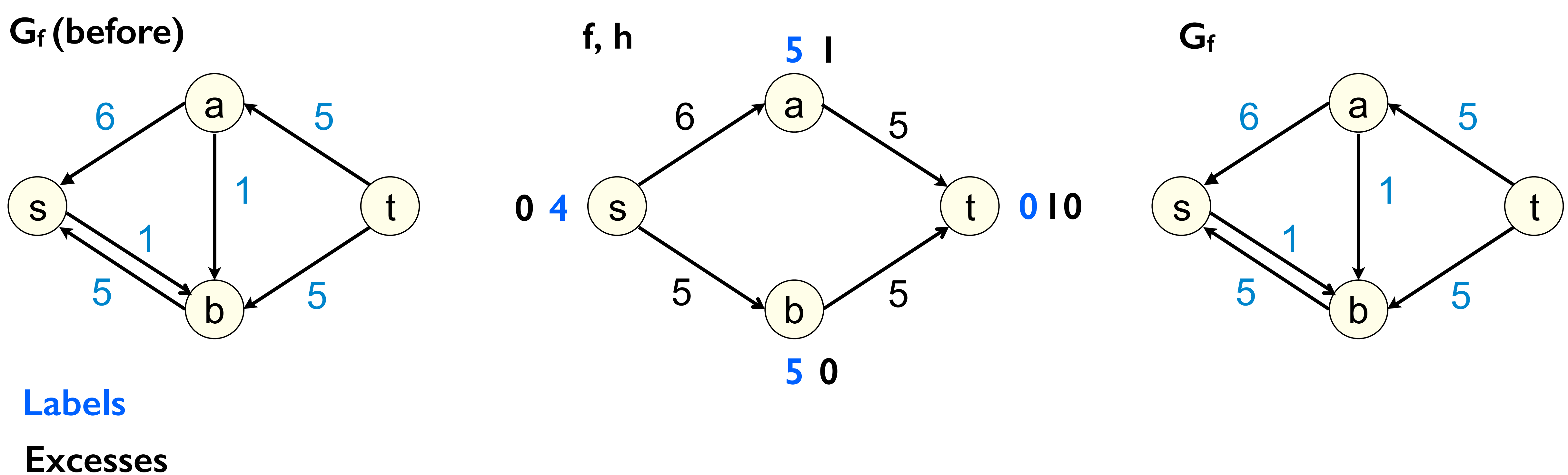
Pre-Flow Push: An Example

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), $c_f(v,w)$)
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1



Pre-Flow Push: An Example

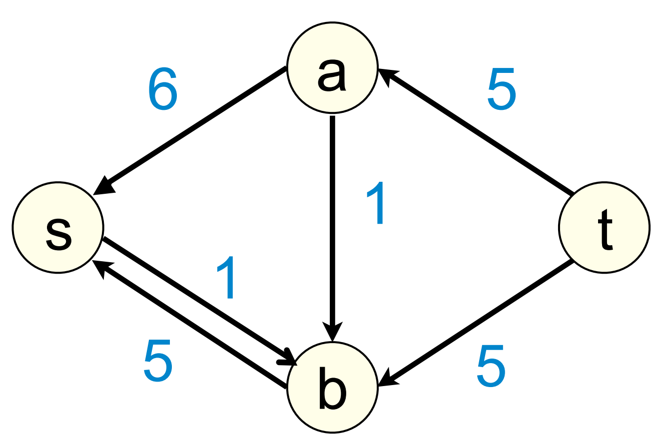
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

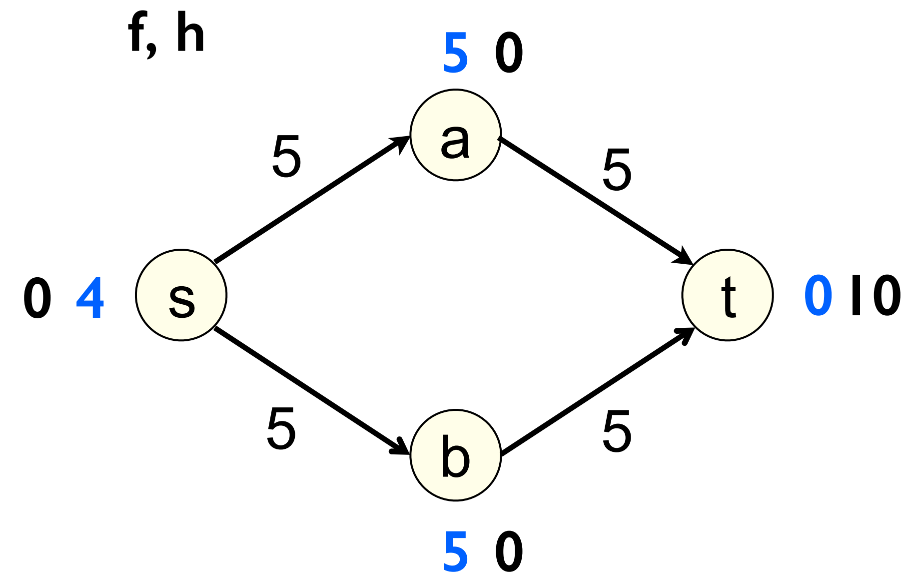
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

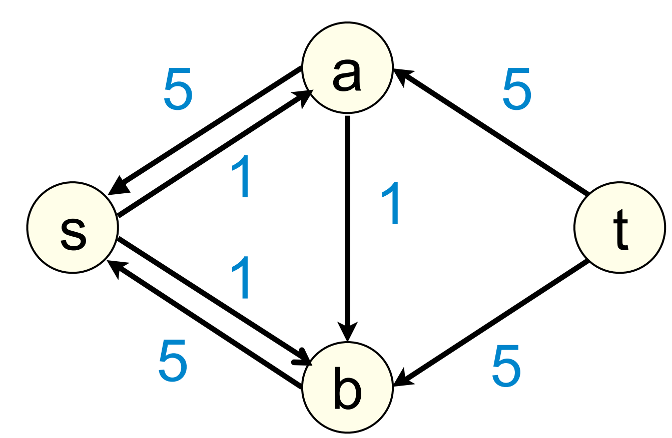
G_f (before)



f, h



G_f



Labels

Excesses

Pre-Flow Push

- Algorithm
- Correctness
- Running Time Analysis

Correctness: Proof Outline

Three Steps:

- Compatibility: Show that the preflow f and the labeling h maintained by the algorithm always obeys a **compatibility** property

- If a flow f is compatible with some labeling, then f is a max-flow

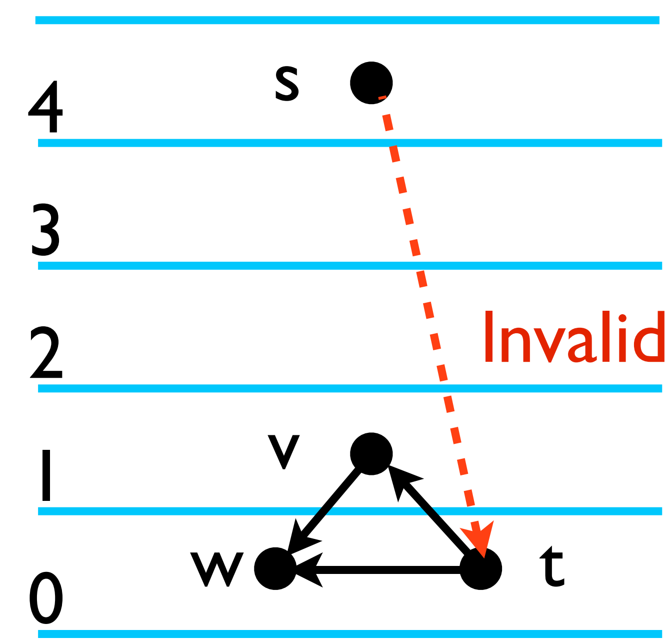
- Preflow-push outputs a flow on termination

Correctness: Compatible Pre-Flows

Preflow: A function f: E $\rightarrow$ R is a preflow if:
1. **Capacity Constraints:** 0 <= f(e) <= c(e)
2. Instead of conservation constraints:

$$\sum_{e \; into \; v} f(e) - \sum_{e \; out \; of \; v} f(e) \geq 0$$

Excess(v) = $\sum_{e \; into \; v} f(e) - \sum_{e \; out \; of \; v} f(e)$



Preflow f and labeling h are compatible if:
1. h(s) = n, h(t) = 0
2. For all edges (v, w) in the residual graph G_f, h(v) <= h(w) + 1

PreFlow Push: Correctness

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

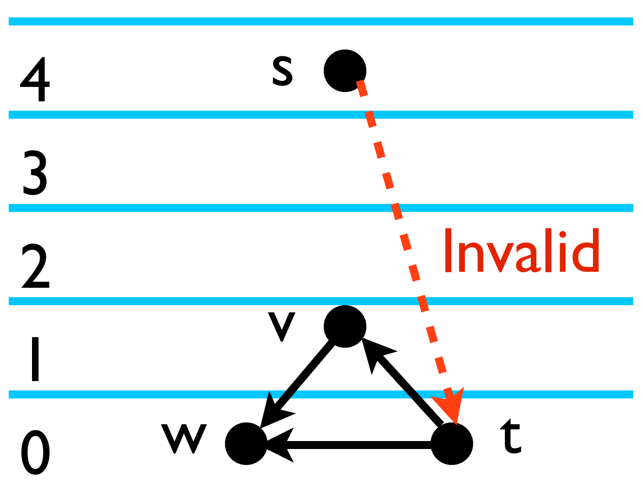
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), $c_f(v,w)$)
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

Preflow f and labeling l are **compatible** if:
1. $h(s) = n$, $h(t) = 0$
2. For all edges (v, w) in G_f, $h(v) \le h(w) + 1$

Invariant: Preflow f and labeling h are always compatible over the Preflow-Push algorithm

Proof: By induction. Initially, compatible, as G_f has no (s, v) edges
Suppose f and h are compatible at time t. At time t+1:
- **Relabel:** Labels increase only if no downward edges in G_f
- **Push:** Edges in G_f may be reversed. If so, as we push from high to low h a downwards edge will become an upwards edge



Correctness: Proof Outline

Three Steps:

- Compatibility: Show that the preflow f and the labeling h maintained by the algorithm always obeys a **compatibility** property

- If a flow f is compatible with some labeling, then f is a max-flow

- Preflow-push outputs a flow on termination

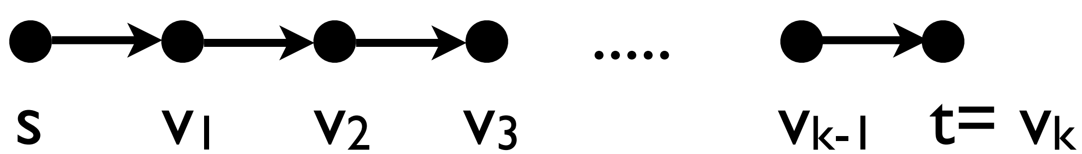
Properties of Compatible PreFlows

Preflow f and labeling h are compatible if:

1. $h(s) = n$, $h(t) = 0$
2. For all edges (v, w) in the residual graph G_f, $h(v) \le h(w) + 1$

Property 1: If preflow f and labeling h are compatible, then there is no s-t path in G_f

Proof: Suppose there is an s-t path in G_f



Due to compatibility,

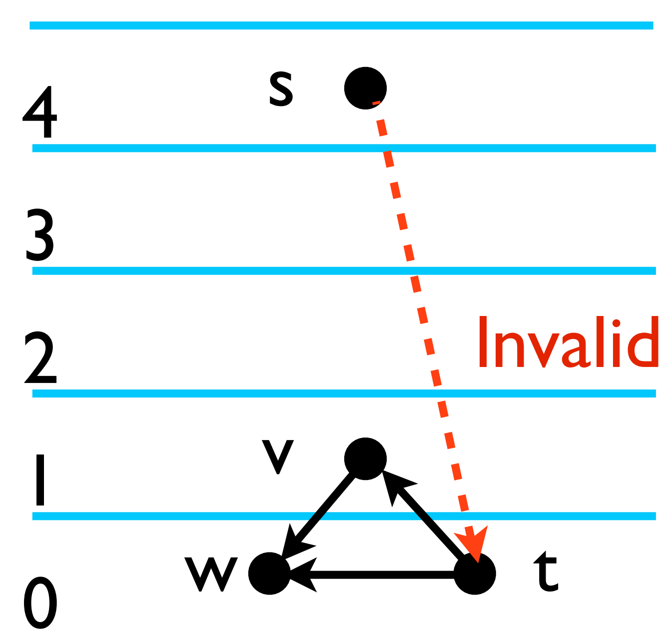
$h(v_1) >= h(s) - 1 = n - 1$

$h(v_2) >= h(v_1) - 1 >= n - 2$

...

$h(t) = h(v_k) - 1 >= n - k > 0$ (as $k < n$)

Contradiction!



Properties of Compatible PreFlows

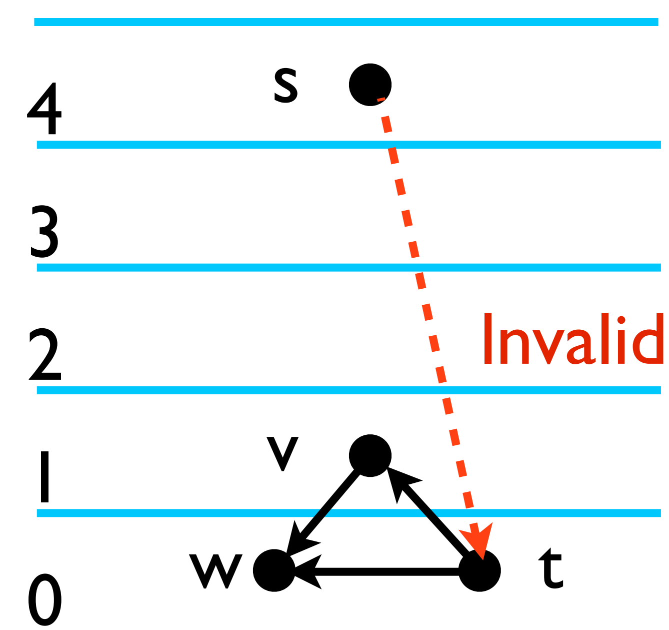
Preflow f and labeling h are compatible if:

1. h(s) = n, h(t) = 0
2. For all edges (v, w) in the residual graph G_f, h(v) <= h(w) + 1

Property 1: If preflow f and labeling h are compatible, then there is no s-t path in G_f

Property 2: If **flow** f and labeling h are compatible, then f is a max flow

Proof: From Property 1 and properties of max flow



Correctness: Proof Outline

Three Steps:

- Compatibility: Show that the preflow f and the labeling h maintained by the algorithm always obeys a **compatibility** property

- If a flow f is compatible with some labeling, then f is a max-flow

- Preflow-push outputs a flow on termination

PreFlow Push: Correctness

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), $c_f(v,w)$)
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

Invariant: Preflow f and labeling h are always compatible over the Preflow-Push algorithm

Fact: When Preflow-push stops, f is a **flow**

Proof: Why does Preflow-push stop?
- No valid push or relabel operation:
  We can always relabel or push if excess(v) > 0 for some v
- No node v with excess(v) > 0:
  Then f is a flow!

Preflow f and labeling h are compatible if:
1. h(s) = n, h(t) = 0
2. For all edges (v, w) in G_f, h(v) <= h(w) + 1

Property 1: If preflow f and labeling h are compatible, then there is no s-t path in G_f

Property 2: If **flow** f and labeling h are compatible, then f is a max flow

PreFlow Push: Correctness

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, $h(w) < h(v)$
  $q = \min(\text{excess}(v), c_f(v,w))$
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
  Increase h(v) by 1

Invariant: Preflow f and labeling h are always compatible over the Preflow-Push algorithm

Fact: When Preflow-push stops, f is a **flow**

From **Property 2** of compatible flows, and **Invariant**, f is a max flow

Thus, Preflow-Push correctly outputs a maxflow

Preflow f and labeling h are compatible if:
1. $h(s) = n$, $h(t) = 0$
2. For all edges (v, w) in G_f, $h(v) <= h(w) + 1$

Property 1: If preflow f and labeling h are compatible, then there is no s-t path in G_f

Property 2: If **flow** f and labeling h are compatible, then f is a max flow

Correctness: Proof Outline

Three Steps:

- Compatibility: Show that the preflow f and the labeling h maintained by the algorithm always obeys a **compatibility** property

- If a flow f is compatible with some labeling, then f is a max-flow

- Preflow-push outputs a flow on termination

Pre-Flow Push

- Algorithm
- Correctness
- Running Time Analysis

Running Time Analysis: Outline

1. How many Push Ops? Relabel Ops?

2. How to implement Push and Relabel Ops efficiently?

Running Time Analysis: Outline

1. How many Relabel Ops?

   Main Idea: Bound the maximum value of h(v) for any node v, and bound #relabel ops through this

Preflow Push: #Relabels

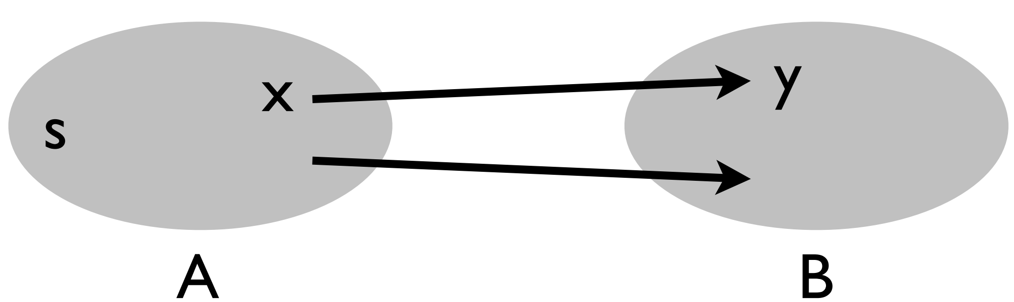
Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), $c_f(v,w)$)
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

Property 1: In a preflow f, if excess(v) > 0, then there is a path from v to s in G_f



A = all nodes v s.t. s is reachable from v in G_f
B = remaining nodes

Fact: Any e=(x, y) from A to B has f(x,y) = 0
If not, (y, x) is in G_f, so there is a y - s path

Now, total excess of nodes in B =

$$\sum_{v \in B} \sum_{e \text{ into } B} f(e) - \sum_{v \in B} \sum_{e \text{ out of } B} f(e) \geq 0$$

Three types of edges e in the sum:

1. Both endpoints of e are in B: f(e) cancels out
2. e = (u, v), u in A, v in B: f(e) = 0
3. e = (v, u), u in A, v in B

Total excess of nodes in B: $-\sum_{v \in B} \sum_{u \in A} f(v, u) \geq 0$

As excess(v) is never <0, excess(v)=0 for v in B

Preflow Push: #Relabels

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, $h(w) < h(v)$
  $q = \min(\text{excess}(v), c_f(v,w))$
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
  Increase h(v) by 1

Compatibility of f and h:
1. $h(s) = n$, $h(t) = 0$
2. For all edges (v, w) in G_f,

Property 1: In a preflow f, if excess(v) > 0, then there is a path from v to s in G_f

Property 2: At any point, for any v, $h(v) <= 2n - 1$

Proof: If excess(v) > 0, there is a v-s path in G_f
Let $v = v_1, \ldots, v_k = s$ be the path
By compatibility:
$h(s) = n, h(v_{k-1}) <= n + 1, h(v_1) <= n+k-1 <= 2n - 1$

If excess(v) = 0, then h(v) has not changed since the last time v had excess > 0

Thus, $h(v) <= 2n - 1$ also

Preflow Push: #Relabels

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, $h(w) < h(v)$
    $q = \min(\text{excess}(v), c_f(v,w))$
    Add q to f(v, w)

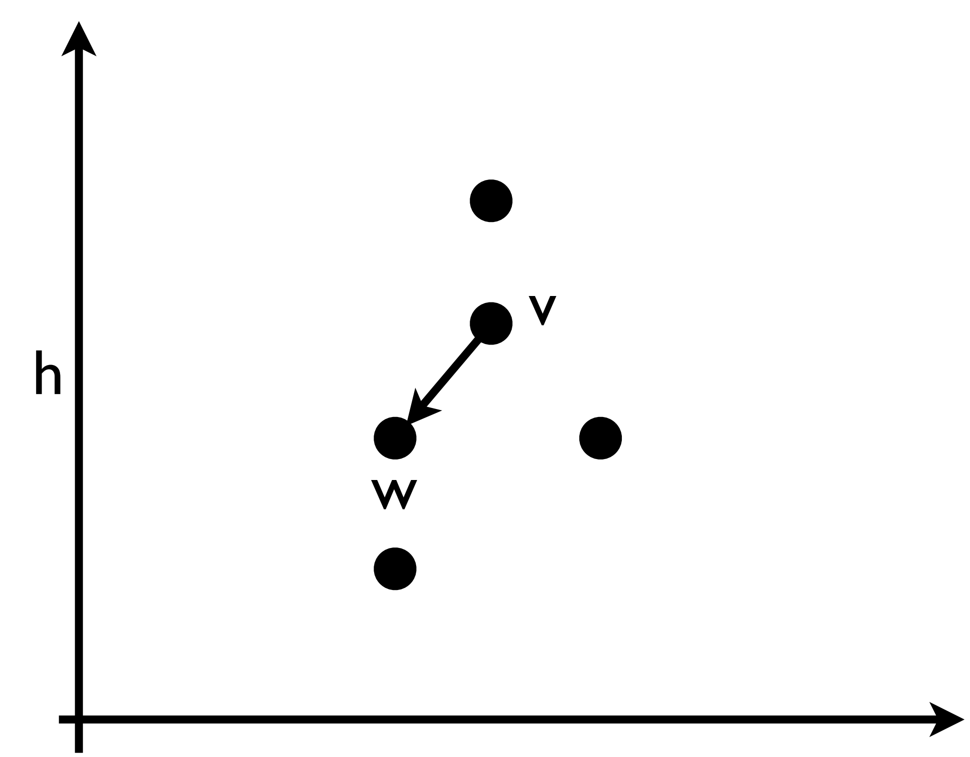
Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
    Increase h(v) by 1

Property 1: In a preflow f, if excess(v) > 0, then there is a path from v to s in G_f

Property 2: At any point in the algorithm, for any v, $h(v) <= 2n - 1$

Property 3: Any node can be relabeled at most 2n times in the algorithm

Proof: Labels never decrease, start at 0, increase by at least 1 per relabel, and can only go upto 2n - 1



Preflow Push: #Relabels

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, $h(w) < h(v)$
  q = min(excess(v), $c_f(v,w)$)
  Add q to f(v, w)

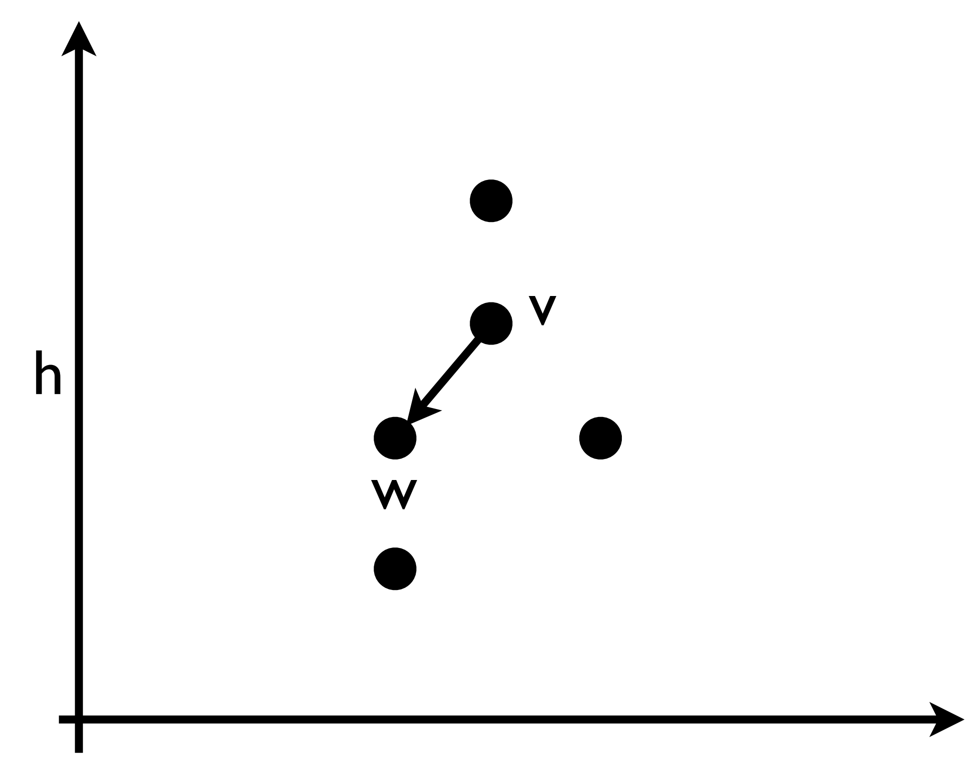
Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
  Increase h(v) by 1

Property 1: In a preflow f, if excess(v) > 0, then there is a path from v to s in G_f

Property 2: At any point in the algorithm, for any v, $h(v) <= 2n - 1$

Property 3: Any node can be relabeled at most 2n times in the algorithm

Total #relabel operations = $O(n^2)$



Running Time Analysis: Outline

1. How many Push Ops? Relabel Ops?

2. How to implement Push and Relabel Ops efficiently?

Running Time Analysis: Outline

1. How many Relabel Ops? How many Push Ops?

   Two types of Push Ops:

   Saturating Pushes: (v, w) is saturated after push(v, w)
   Same edge can't be pushed on until a relabel (we will see why!)

   Non-saturating Pushes: excess(v) = 0 after push(v, w)

Preflow Push: #Pushes

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

Two kinds of Pushes:
Saturating: (v,w) is not in G_f after push
Nonsaturating: excess(v) becomes 0 after push

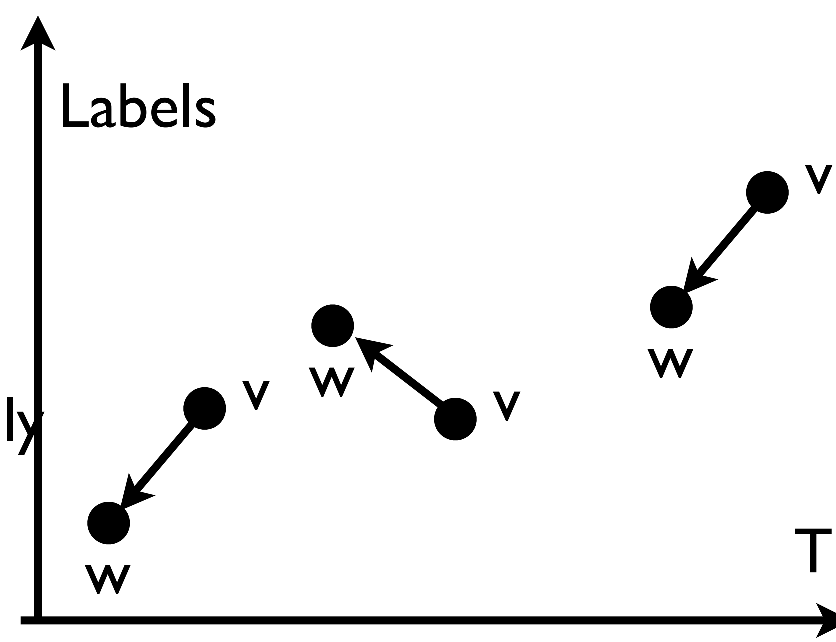
Property 1: There are at most 2mn saturating pushes

Proof: For a fixed edge (v,w), after a saturating push, we can only push along (v, w) again once v is relabeled

#relabels of v <= 2n

#saturating pushes along (v, w) <= 2n

#saturating pushes along all m edges <= 2nm



(v,w) disappears from G_f after saturating push, appears only after w to v push

Running Time Analysis: Outline

1. How many Relabel Ops? How many Push Ops?

   Two types of Push Ops:

   Saturating Pushes: (v, w) is saturated after push(v, w)
   Same edge can't be pushed on until a relabel

   Non-saturating Pushes: excess(v) = 0 after push(v, w)
   Harder to bound. Need to use a potential function argument

Preflow Push: #Pushes

Two kinds of Pushes:
Saturating: (v,w) is not in G_f after push
Nonsaturating: excess(v) becomes 0 after push

Property 0: There are <= $2n^2$ relabels
Property 1: There are <= 2mn saturating pushes
Property 2: There are <= $4mn^2$ non-saturating pushes

Proof: Define a potential function G(f, h):

$$G(f,h) = \sum_{v:excess(v)>0} h(v)$$

Initially, G(f, h) = 0

At any time, G(f, h) >= 0

At a relabel operation, G(f, h) can increase by 1

At a saturating push operation, G(f, h) can increase if w gets >0 excess. Total increase = h(w) <= 2n - 1

At a non-saturating push operation, G(f, h) will decrease by h(v), but may increase by h(w) if w gets >0 excess
But h(v) > h(w), so G(f, h) will decrease by at least 1

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
q = min(excess(v), c_f(v,w))
Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
Increase h(v) by 1

Total increase from relabels <= $2n^2$

Total increase from saturating pushes pushes <= 2mn(2n - 1)

(#non-saturating pushes) x 1
<= Total decrease from such pushes
<= total increase from anything else
<= $2n^2$ + 2mn(2n - 1) = $4mn^2$

#Non-saturating Pushes <= $4mn^2$

Preflow Push: #Pushes

Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
    q = min(excess(v), c_f(v,w))
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
    Increase h(v) by 1

Two kinds of Pushes:
Saturating: (v,w) is not in G_f after push
Nonsaturating: excess(v) becomes 0 after push

Property 0: There are at most $2n^2$ relabels

Property 1: There are at most 2mn saturating pushes

Property 2: There are at most $4mn^2$ non-saturating pushes

Total #pushes: $O(mn^2)$

Running Time Analysis: Outline

1. How many Push Ops? Relabel Ops?

2. How to implement Push and Relabel Ops efficiently?

Preflow Push: Data Structures

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

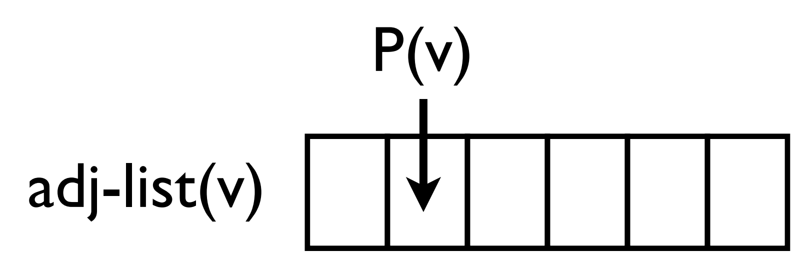
While there is a node (other than t) with positive excess
    Pick a node v with excess(v) > 0
    If there is an edge (v, w) in E_f s. t. push(v, w) applies
        Push(v, w)
    Else
        Relabel(v)

Push(v, w):
Applies if excess(v) > 0, $h(w) < h(v)$
    q = min(excess(v), $c_f(v,w)$)
    Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, $h(w) >= h(v)$
    Increase h(v) by 1

1. For each label, use a list to maintain nodes with excess > 0
   Time to select a v with excess(v) > 0: O(1)
   Time to insert or delete: O(1)

2. For each v, maintain all (v,w) in E_f in an adjacency list
   Keep a pointer P(v) to the next edge we can push on
   If excess(v) = 0, P(v) stays on the current edge
   Move P(v) by 1 when current edge is saturated
   [Recall: If we push(v,w) and saturate it, then, we cannot push(v,w) again until v is relabeled]
   Update P(v) and the list when v is relabeled



Preflow Push: Data Structures

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

1. For each label, use a list to maintain nodes with excess > 0
   Time to select a v with excess(v) > 0: O(1)
   Time to insert or delete: O(1)

2. For each v, maintain all (v,w) in E_f in an adjacency list
   Keep a pointer P(v) to the next edge we can push on
   If excess(v) = 0, P(v) stays on the current edge
   Move P(v) by 1 when current edge is saturated
   [Recall: If we push(v,w) and saturate it, then, we cannot push(v,w) again until v is relabeled]
   Update P(v) and the list when v is relabeled

Time per relabel = O(1)

Time per push = O(1)

Time to maintain list after relabeling v = O(deg(v))

Total running time
= O(m) x #relabels/node +
  O(#pushes + #relabels)
= $O(mn) + O(mn^2) = O(mn^2)$

Running Time Analysis: Outline

1. How many Push Ops? Relabel Ops?
   #pushes = $O(mn^2)$, #relabels = $O(n^2)$

2. How to implement Push and Relabel Ops efficiently?
   Data structure which takes: $O(1)$ per push, $O(deg(v))$ to relabel v once
   Total running time = $O(mn^2)$