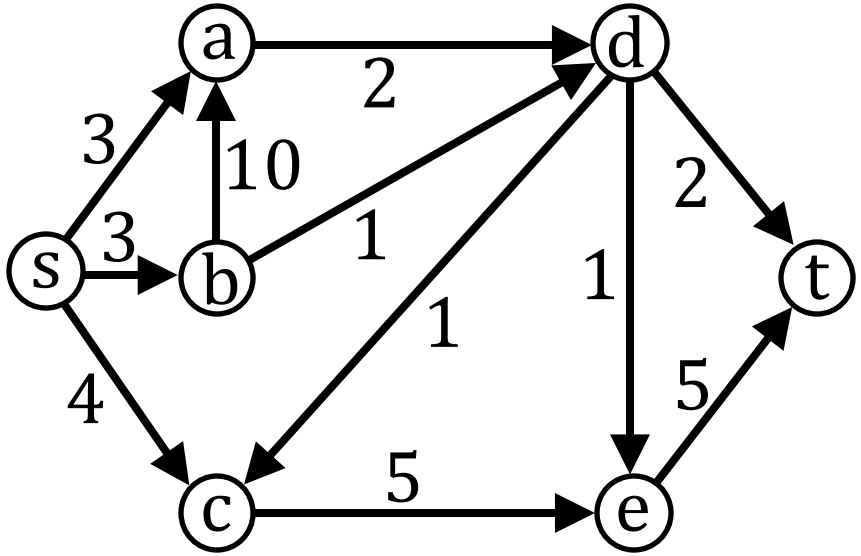
Flow Networks
CSCI 432

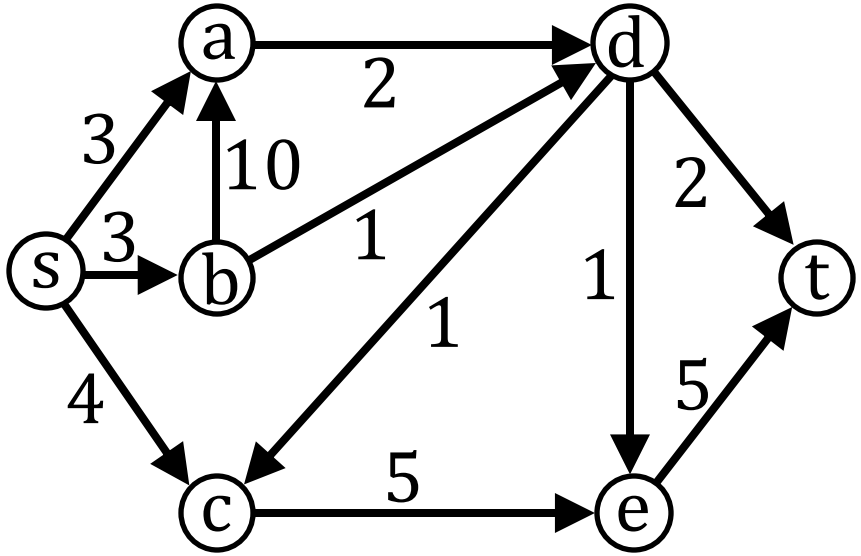
Motivation

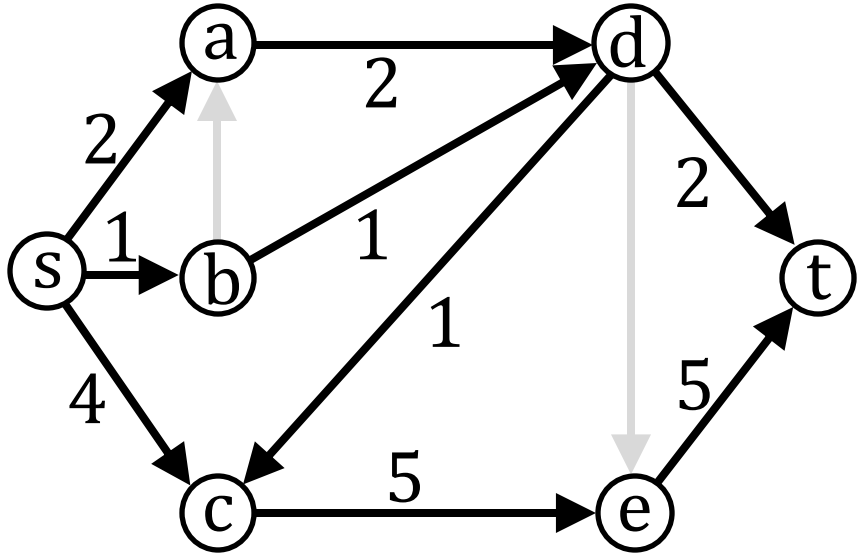
Suppose we have a directed graph that represent an oil pipeline network. Edge weight represent pipe capacity. How much oil can we transfer from source s to sink t?



Motivation

Suppose we have a directed graph that represent an oil pipeline network. Edge weight represent pipe capacity. How much oil can we transfer from source s to sink t?

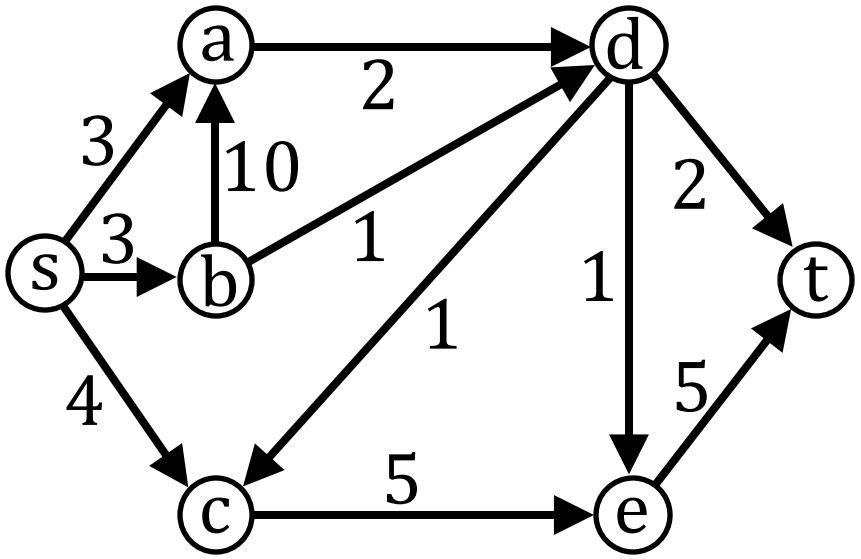




Total flow = 7

Motivation

Suppose we have a directed graph that represent an oil pipeline network. Edge weight represent pipe capacity. How much oil can we transfer from source s to sink t?



Flow networks are commonly used to model transportation networks:

- Edges carry traffic (capacity constrained).
- Nodes act as junctions between edges.

Examples:

- Internet routing.
- Road networks.
- Electricity distribution.

Flow Network

Flow Network:

An $s-t$ flow is a function f: $E \rightarrow \mathbb{R}^{+}$ such that:

Flow Network

Flow Network:

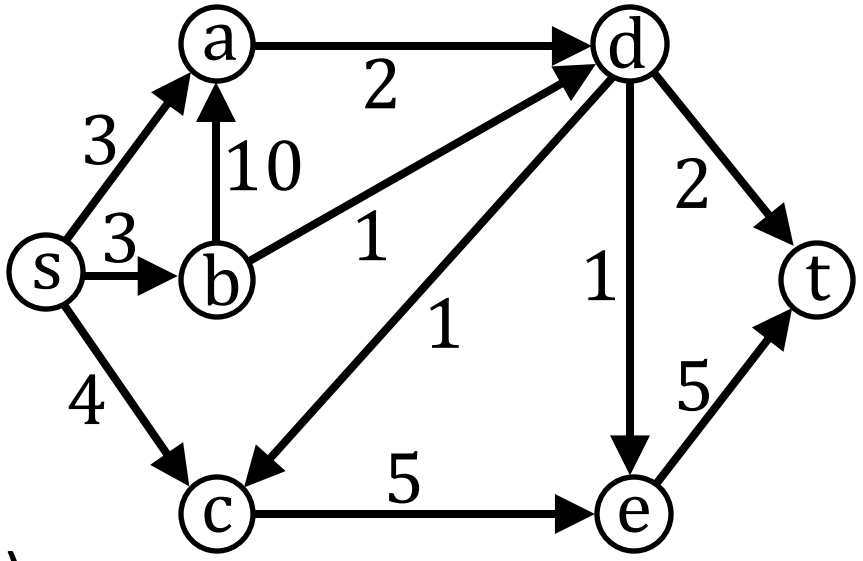
- Directed-edge graph, $G = (V, E)$.
- Non-negative edge capacity, c_e.
- Single source, s, without input edges.
- Single sink, t, without output edges.

An $s - t$ flow is a function f: $E \rightarrow \mathbb{R}^+$ such that:

Flow Network

Flow Network:

- Directed-edge graph, $G = (V, E)$.
- Non-negative edge capacity, c_e.
- Single source, s, without input edges.
- Single sink, t, without output edges.



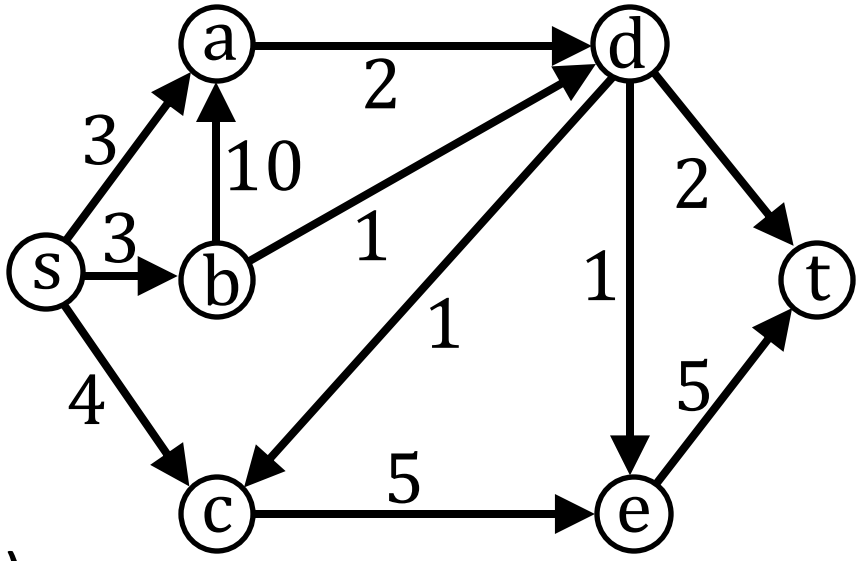
An $s - t$ flow is a function $f\colon E \to \mathbb{R}^+$ such that:

- $0 \leq f(e) \leq c_e, \forall e \in E$. (capacity constraint)

Flow Network

Flow Network:

- Directed-edge graph, $G = (V, E)$.
- Non-negative edge capacity, c_e.
- Single source, s, without input edges.
- Single sink, t, without output edges.



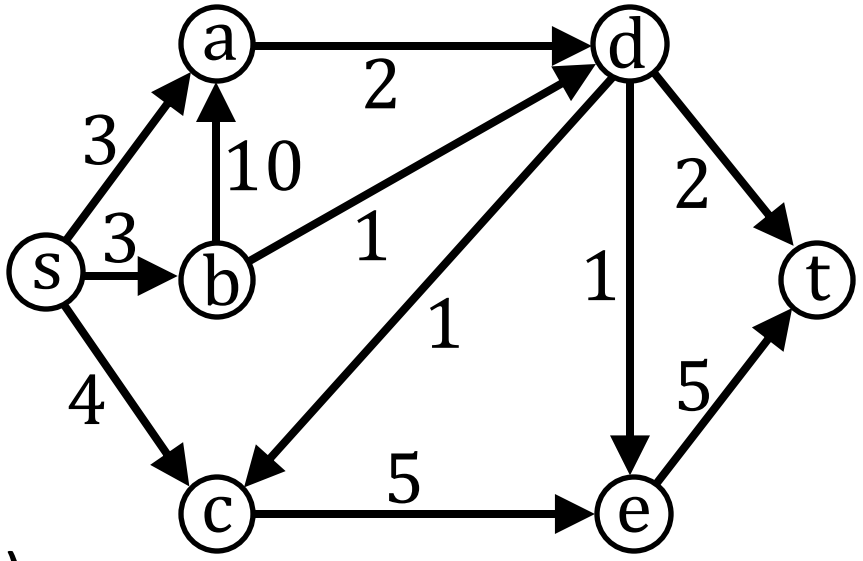
An $s - t$ flow is a function $f\colon E \to \mathbb{R}^+$ such that:

- $0 \leq f(e) \leq c_e, \forall e \in E$. (capacity constraint)
- $\sum_{e \in \text{input}(v)} f(e) = \sum_{e \in \text{output}(v)} f(e)\,, \forall v \in V \setminus \{s, t\}$. (conservation of flow constraint: "Everything that goes into a node has to come out, except for s and t")

Flow Network

Flow Network:

- Directed-edge graph, $G = (V, E)$.
- Non-negative edge capacity, c_e.
- Single source, s, without input edges.
- Single sink, t, without output edges.



An $s - t$ flow is a function $f\colon E \to \mathbb{R}^+$ such that:

- $0 \leq f(e) \leq c_e, \forall e \in E$. (capacity constraint)
- $\sum_{e \epsilon \text{input}(v)} f(e) = \sum_{e \epsilon \text{output}(v)} f(e)\,, \forall v \in V \setminus \{s, t\}$. (conservation of flow constraint: "Everything that goes into a node has to come out, except for s and t")
- Value of flow = $val(f)$ = $\sum_{e \epsilon \text{output}(s)} f(e)$ = $\sum_{e \epsilon \text{input}(t)} f(e)$

Flow Network

Maximum Flow Problem:
Given a flow network, find the maximum possible value of flow.

Flow Network:

- Directed-edge graph, $G = (V, E)$.
- Non-negative edge capacity, c_e.
- Single source, s, without input edges.
- Single sink, t, without output edges.

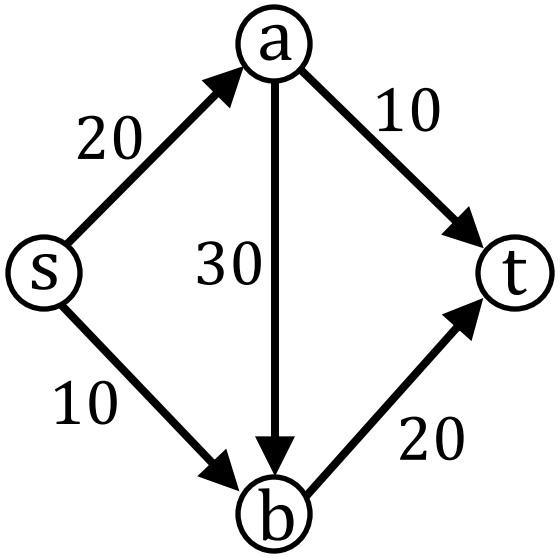
An $s - t$ flow is a function f: $E \rightarrow \mathbb{R}^+$ such that:

- $0 \leq f(e) \leq c_e, \forall e \in E$. (capacity constraint)
- $\sum_{e \epsilon \text{input}(v)} f(e) = \sum_{e \epsilon \text{output}(v)} f(e)\,, \forall v \in V \setminus \{s, t\}$. (conservation of flow constraint: "Everything that goes into a node has to come out, except for s and t")
- Value of flow = $val(f)$ = $\sum_{e \epsilon \text{output}(s)} f(e)$ = $\sum_{e \epsilon \text{input}(t)} f(e)$

Maximum Flow Algorithm

Can we use Dynamic Programming?

Can we use a Greedy approach?



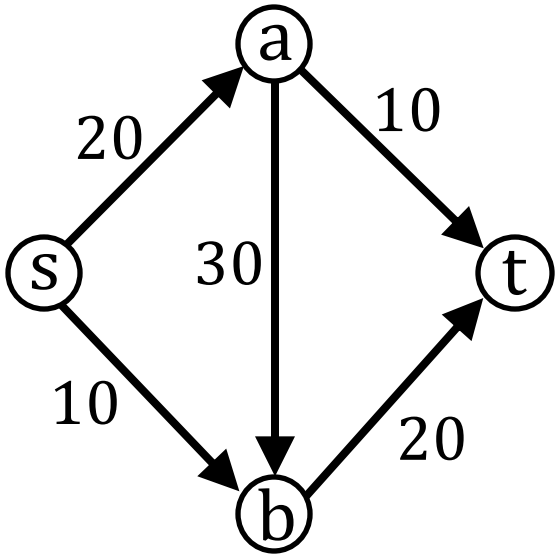
Maximum Flow Algorithm

Can we use Dynamic Programming?

What is the optimal substructure?

Can we use a Greedy approach?

What would the greedy choice be?



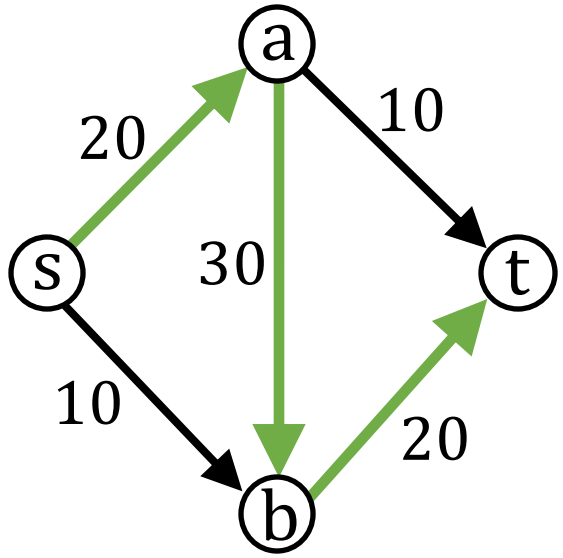
Maximum Flow Algorithm

Can we use Dynamic Programming?

What is the optimal substructure?

Can we use a Greedy approach?

What would the greedy choice be?



Greedy Choice: Select path that can handle most flow, update capacities, repeat.

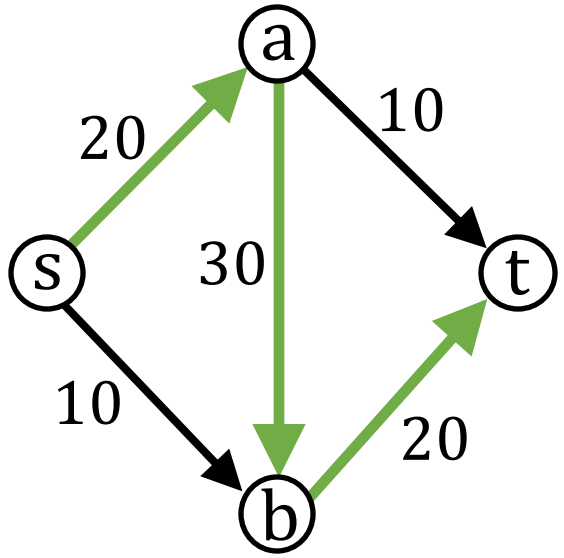
Maximum Flow Algorithm

Can we use Dynamic Programming?

What is the optimal substructure?

Can we use a Greedy approach?

What would the greedy choice be?

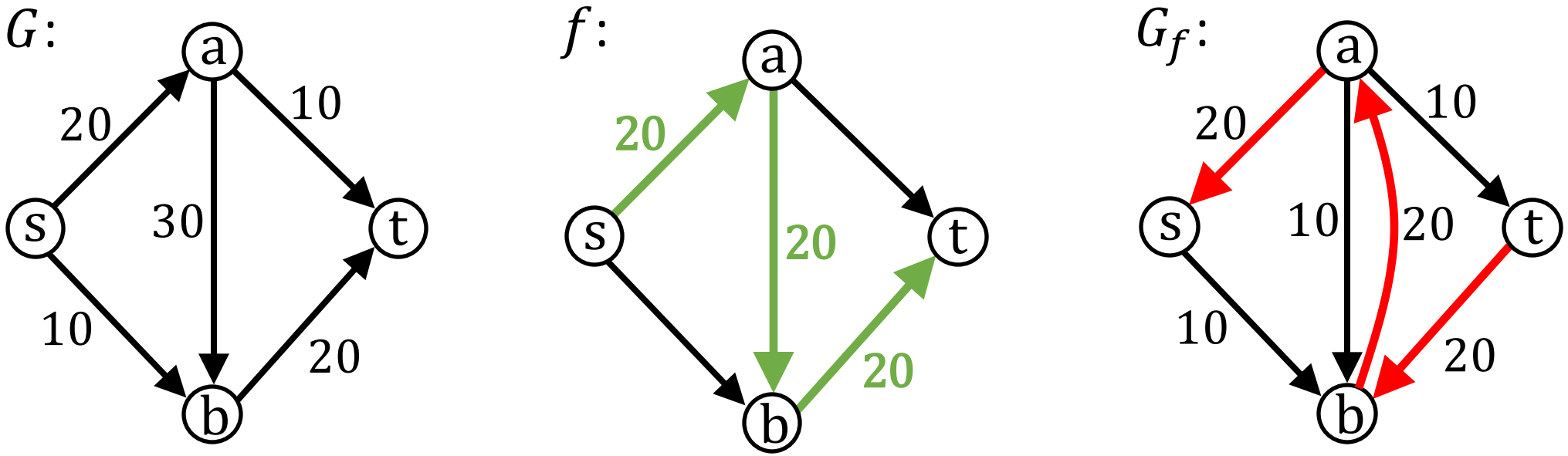


Greedy Choice: Select path that can handle most flow, update capacities, repeat.

Flow of 20, but max flow is 30. Need way to "undo" parts of previous decisions.

Residual Graph

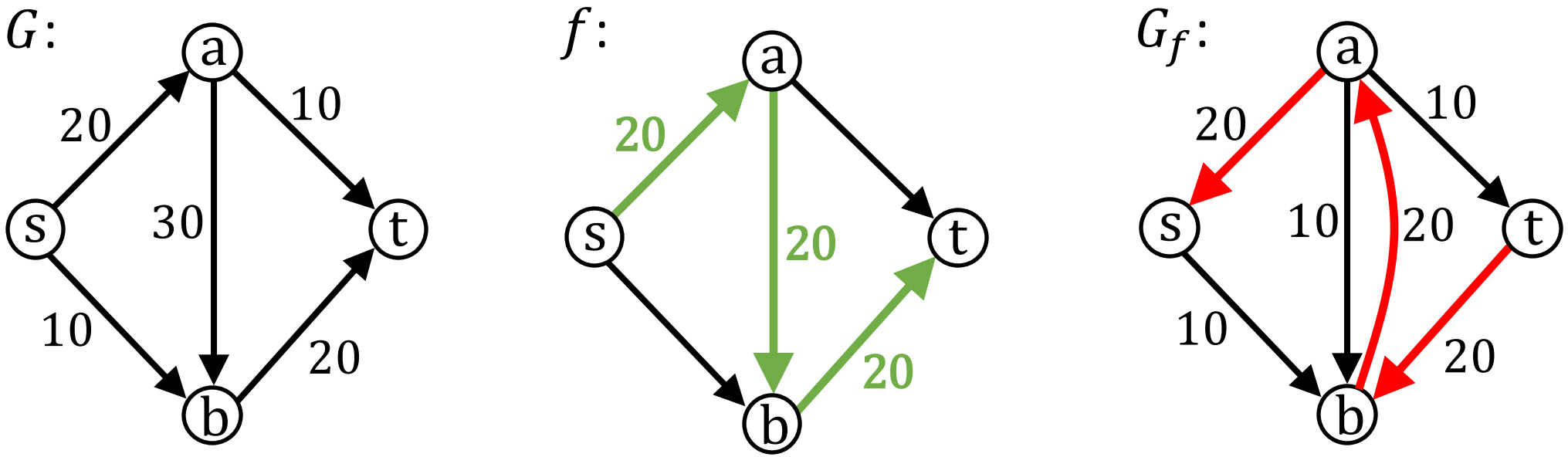
Residual Graph: Tool to re-route flow we already decided to push.



Residual Graph

Given a flow network G, and a flow f, define the residual graph G_f as:

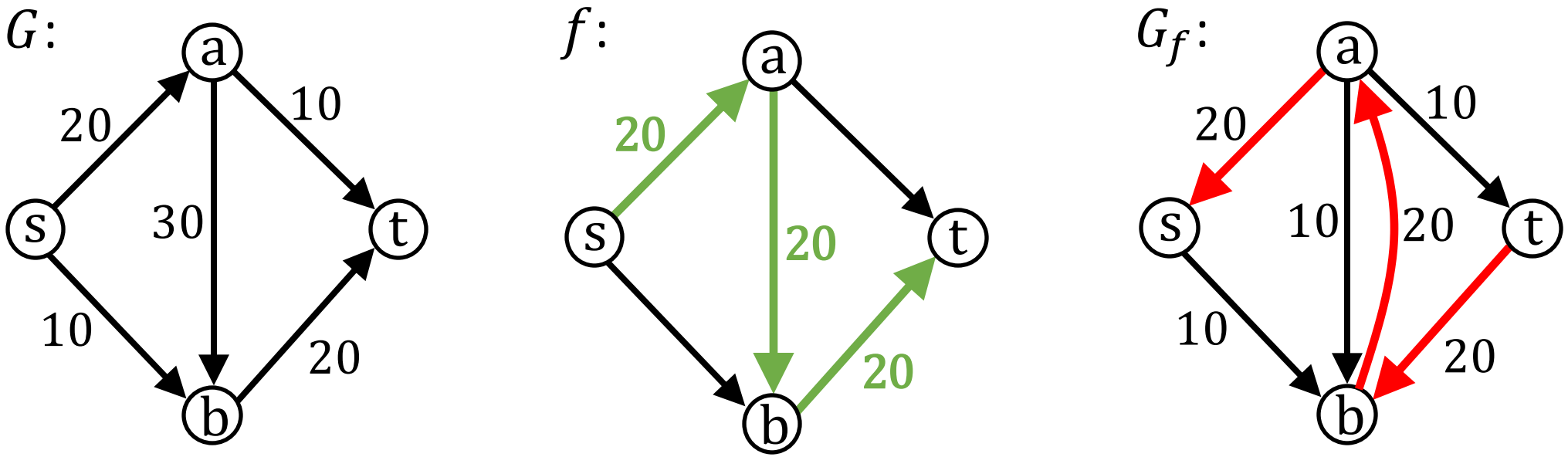
- Identical nodes as G.



Residual Graph

Given a flow network G, and a flow f, define the residual graph G_f as:

- Identical nodes as G.
- For each edge e, if $f(e) < c_e$, let $c_e = c_e - f(e)$.

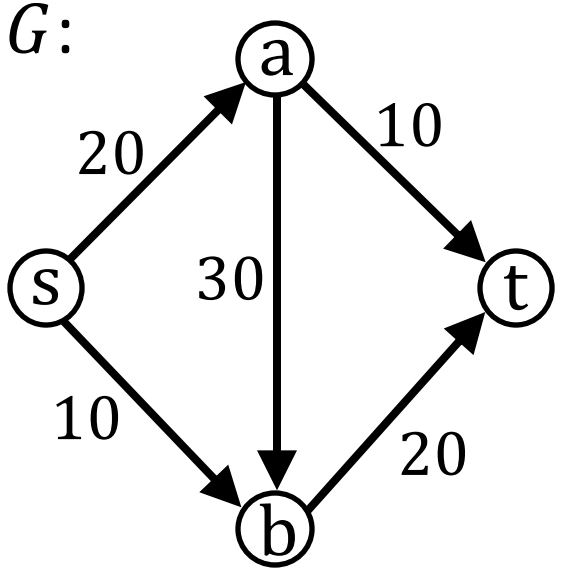


Residual Graph

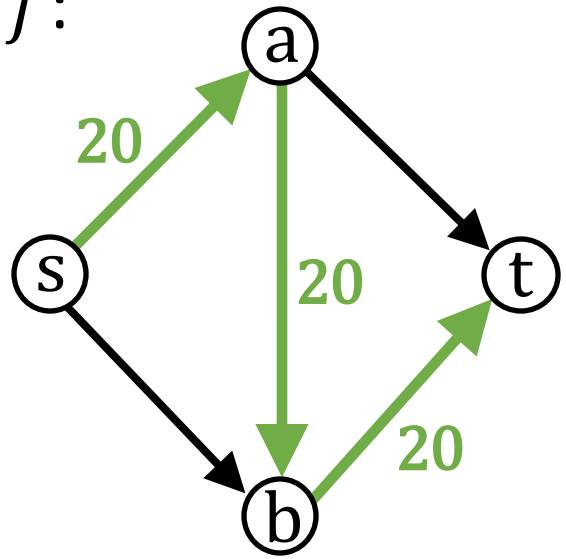
Given a flow network G, and a flow f, define the residual graph G_f as:

- Identical nodes as G.
- For each edge e, if $f(e) < c_e$, let $c_e = c_e - f(e)$.
- For each edge $e = (u, v)$, if $f(e) > 0$, create new edge $e' = (v, u)$ with $c_{e'} = f(e)$ (e' called back edge).

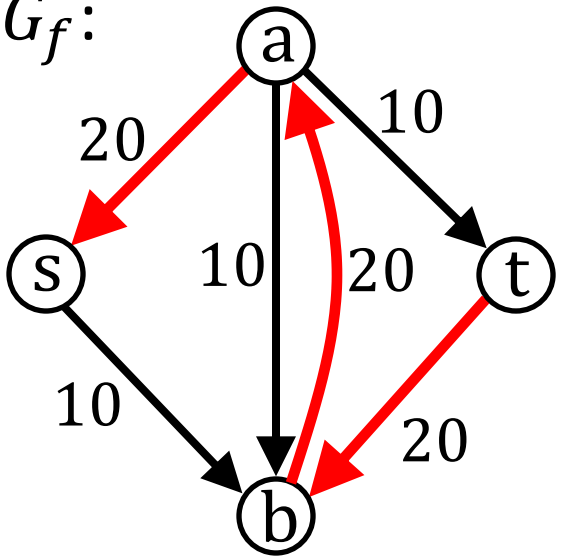
G:



f:

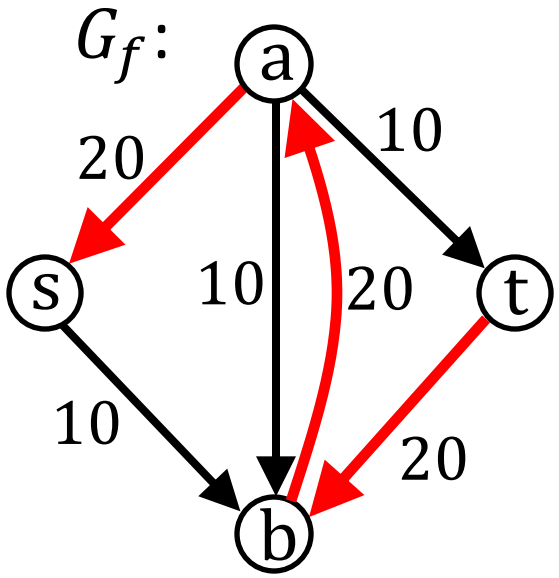


G_f:



Flows in Residual Graphs

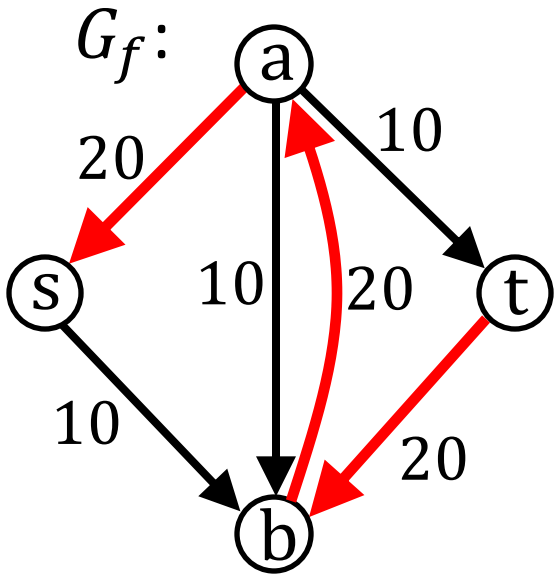
Let P be a simple $s - t$ path in G_f.



$P = ?$

Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

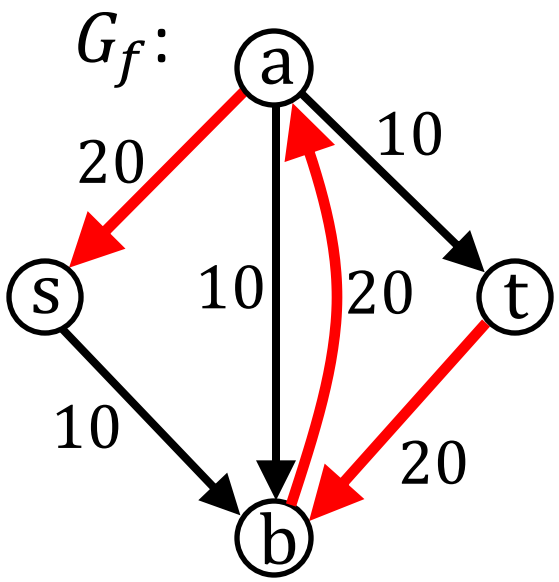


$$P = s \to b \to a \to t$$

Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.



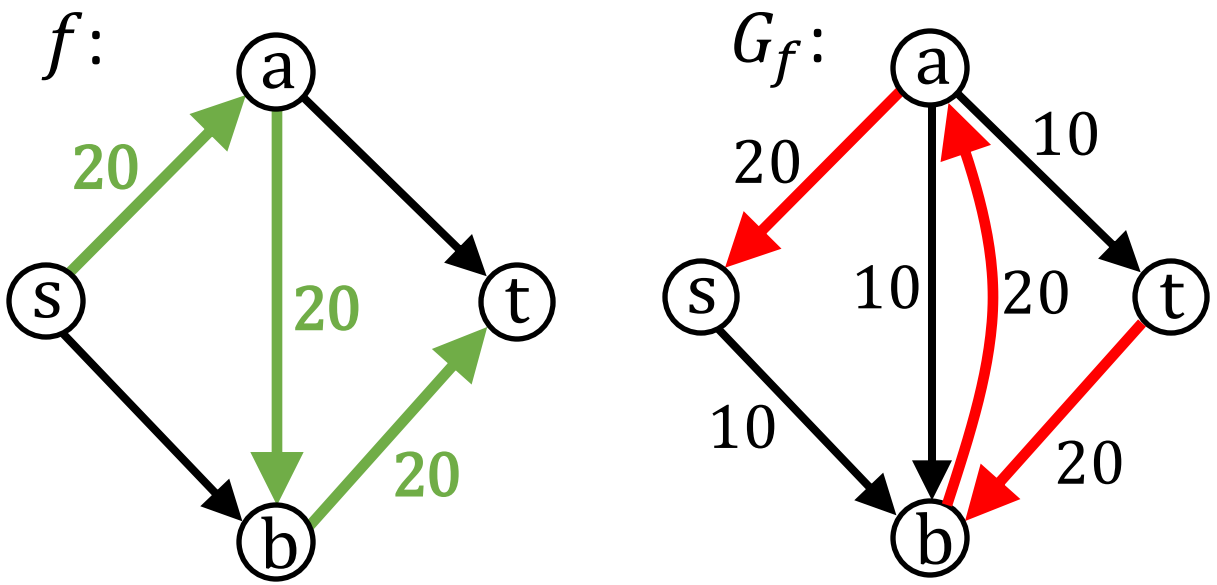
$$P = s \to b \to a \to t$$

Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```



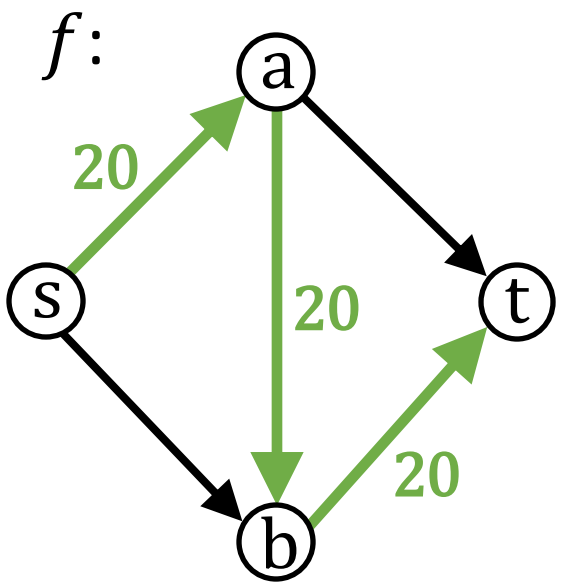
$P = s \to b \to a \to t$

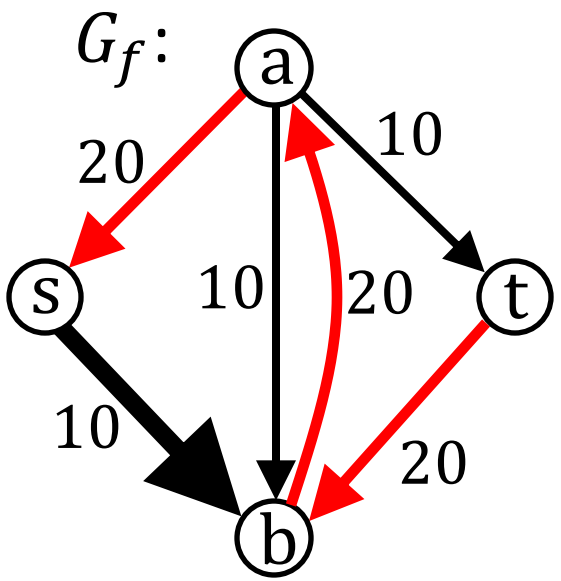
Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```





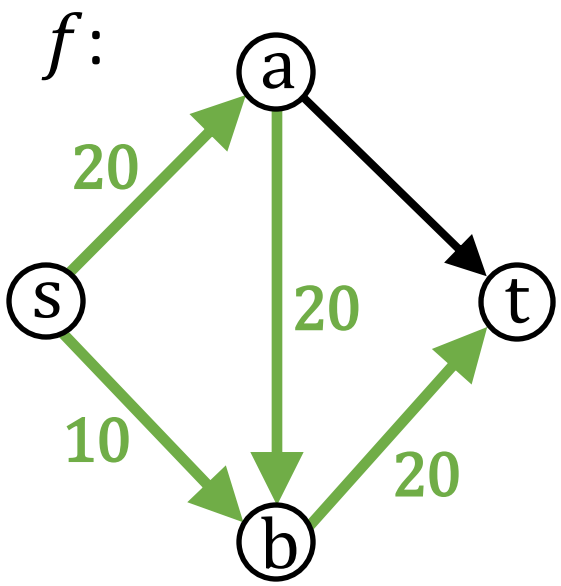
$$P = s \to b \to a \to t$$

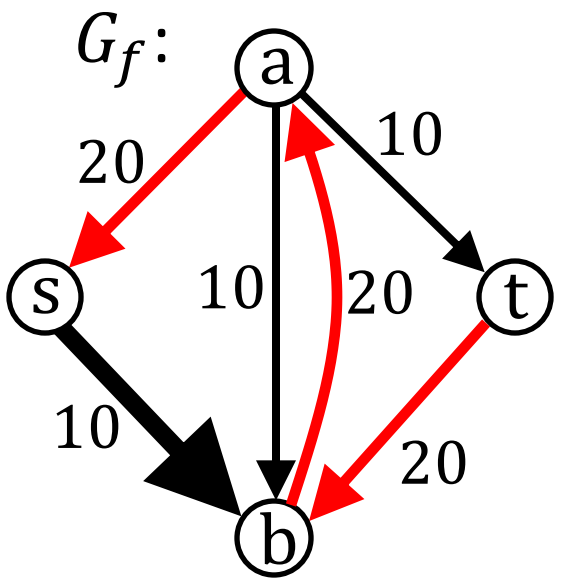
Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```





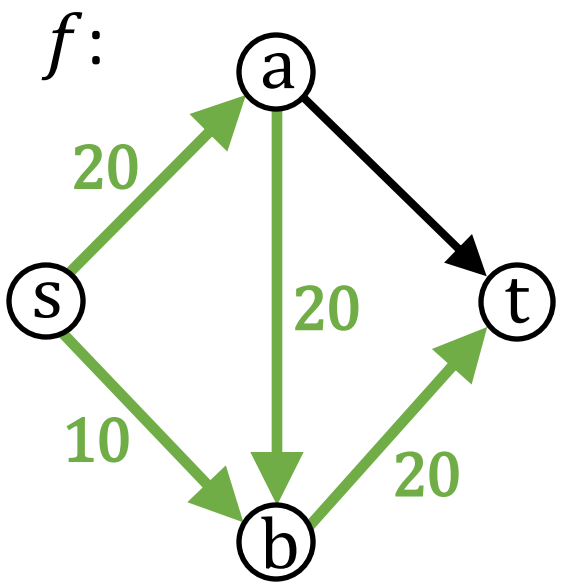
$$P = s \to b \to a \to t$$

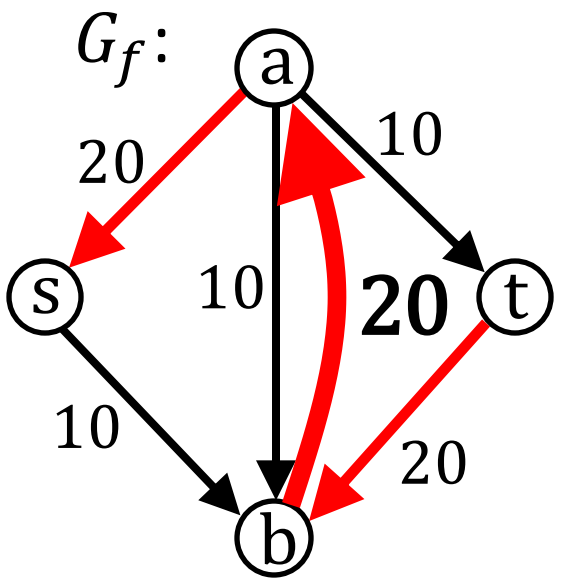
Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```





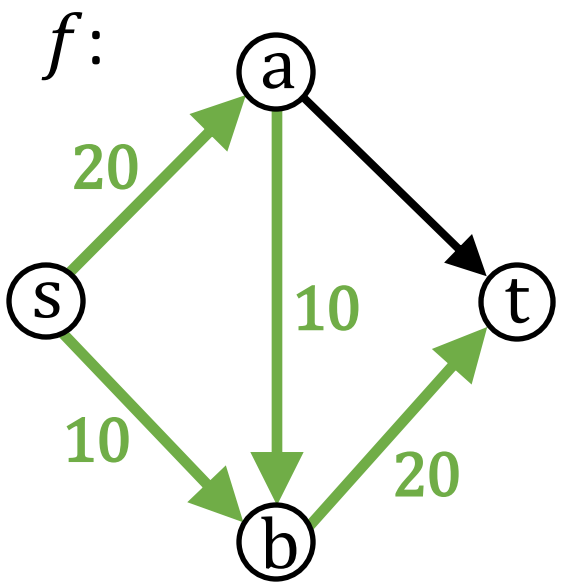
$$P = s \to b \to a \to t$$

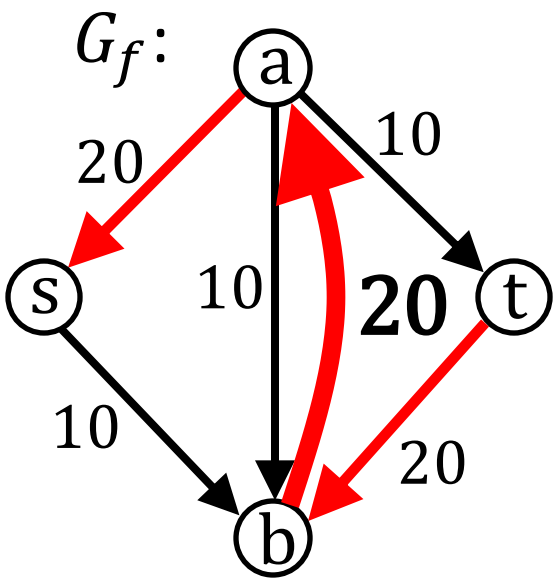
Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```





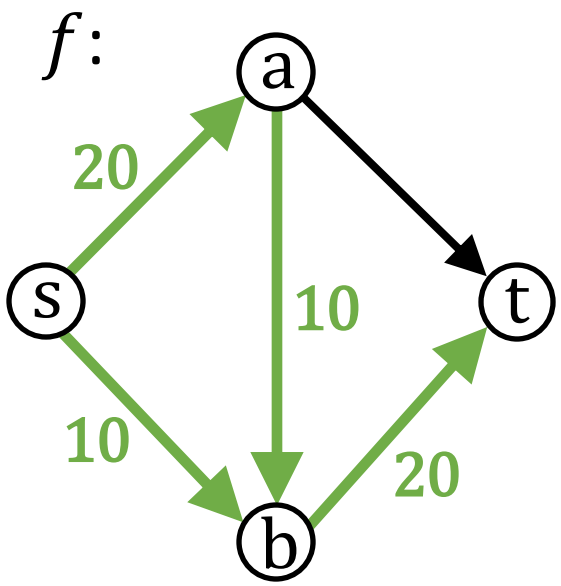
$P = s \to b \to a \to t$

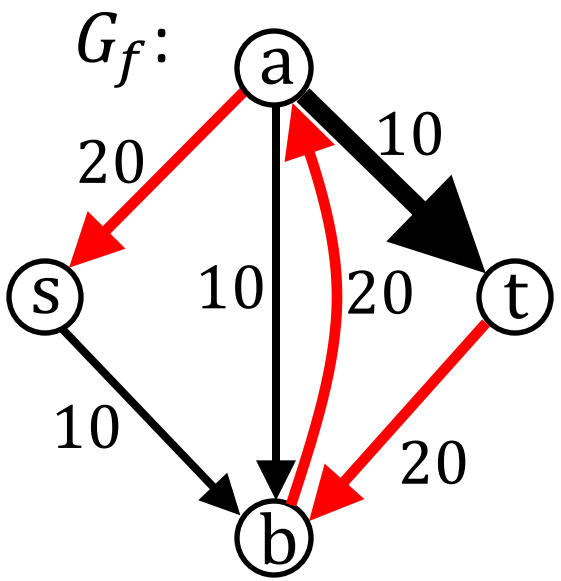
Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```





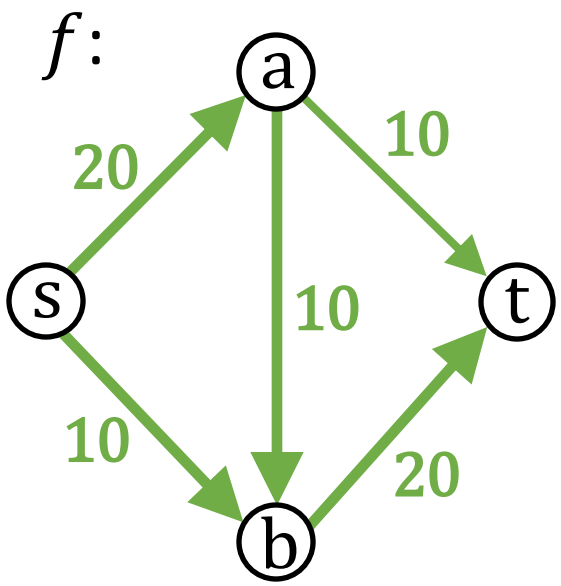
$P = s \to b \to a \to t$

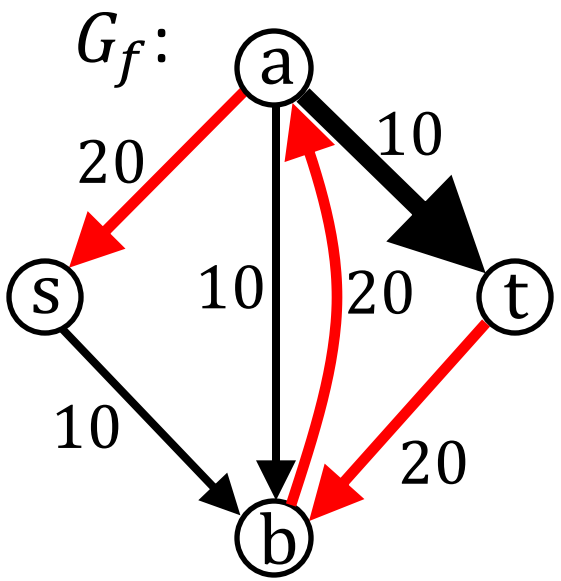
Flows in Residual Graphs

Let P be a simple $s - t$ path in G_f.

`bottleneck(P,f)` = minimum residual capacity on any edge in P = 10.

```
augment(f, P)
   b = bottleneck(P,f) = 10
   for each edge (u, v) in P
      if (u, v) is a back edge
         f((v, u)) -= b
      else
         f((u, v)) += b
   return f
```





$$P = s \to b \to a \to t$$

Ford-Fulkerson Algorithm

```
Max-Flow(G)
   f(e) = 0 for all e in G
   while simple s-t path in residual graph Gf exists
      P = simple s-t path in Gf
      f'= augment(f, P)
      f = f'
      Gf = Gf'
   return f
```

I.e. Push flow along P.

Recalculate residual graph based on new flow f'.