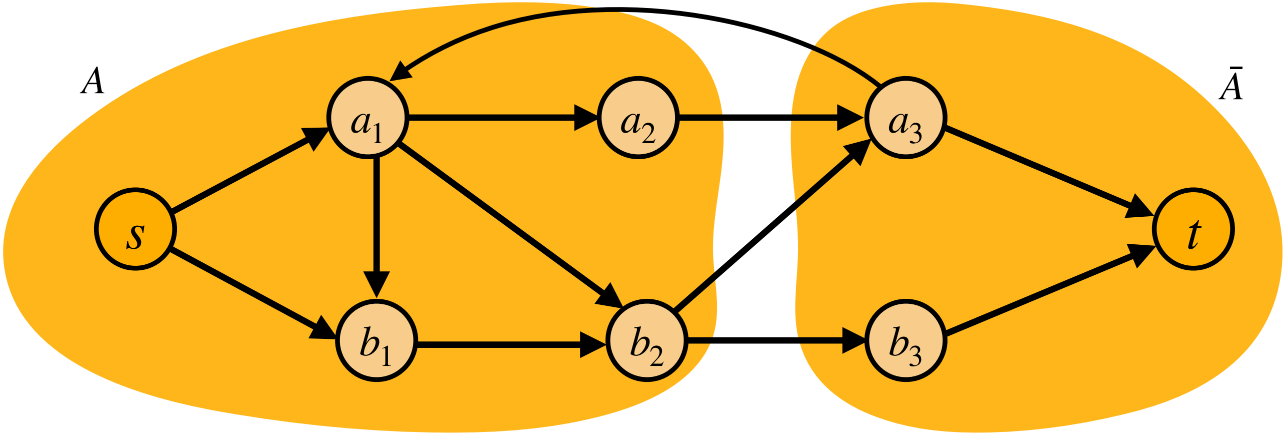
COL 351: Analysis and Design of Algorithms

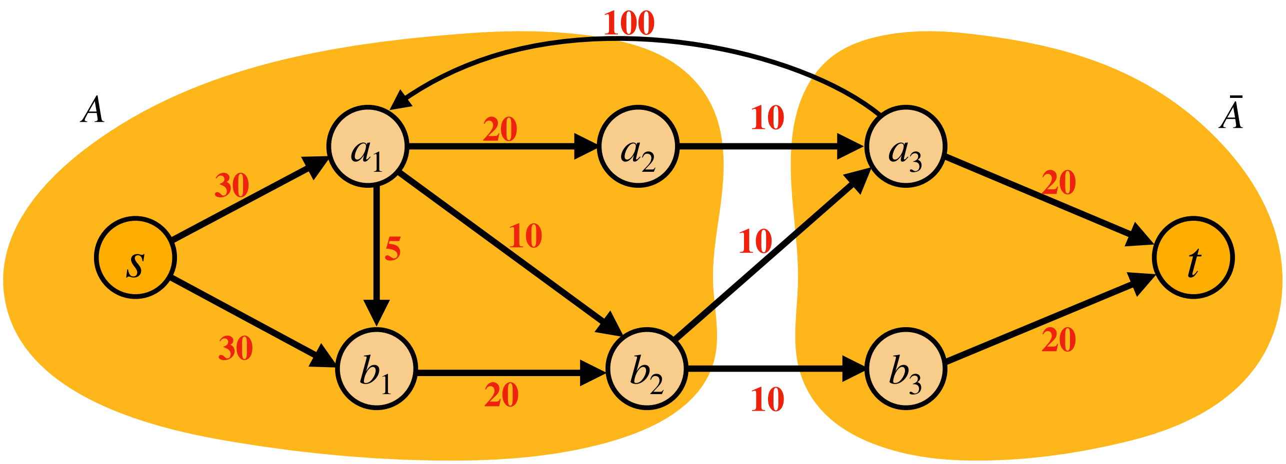
Lecture 32

(s,t)-Cuts



Definition: Any partition $(A, \bar{A})$ of vertices satisfying $s \in A$, $t \in \bar{A}$.

Definitions



For any cut $(A, \bar{A})$,

$c(A, \bar{A}) = \sum_{\substack{(x,y) \in \\ (A \times \bar{A}) \cap E}} c(x,y)$	$f_{out}(A) = \sum_{\substack{(x,y) \in \\ (A \times \bar{A}) \cap E}} f(x,y)$	$f_{in}(A) = \sum_{\substack{(x,y) \in \\ (\bar{A} \times A) \cap E}} f(x,y)$

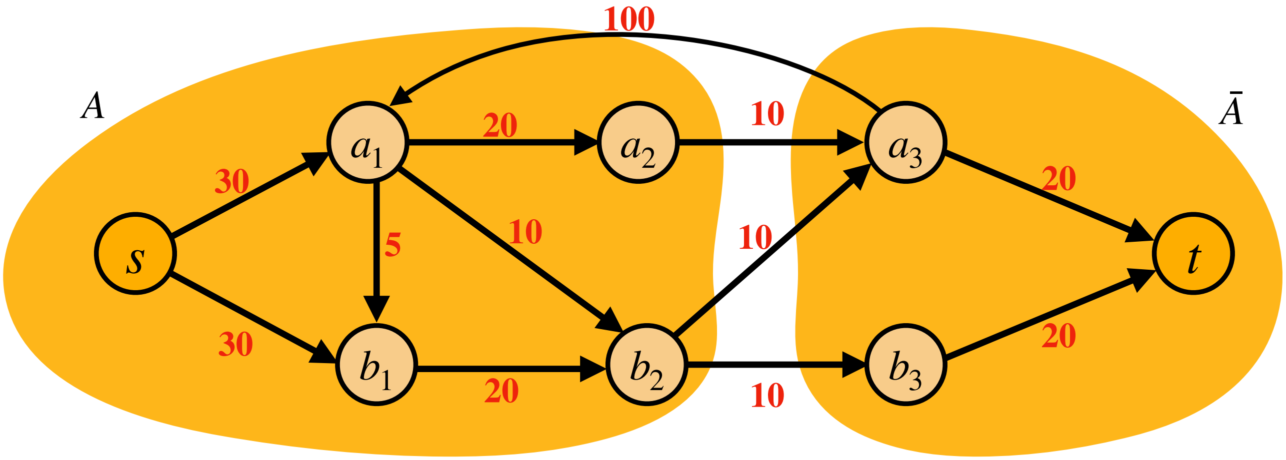
Eg. $10 + 10 + 10 = 30$

Eg. ≤ 30

Eg. ≤ 30

≤ 100

(s,t)-Min-Cut



Definition:

A cut $(A, \bar{A})$ with $s \in A,\ t \in \bar{A}$ for which $c(A, \bar{A})$, i.e. capacity, is minimised.

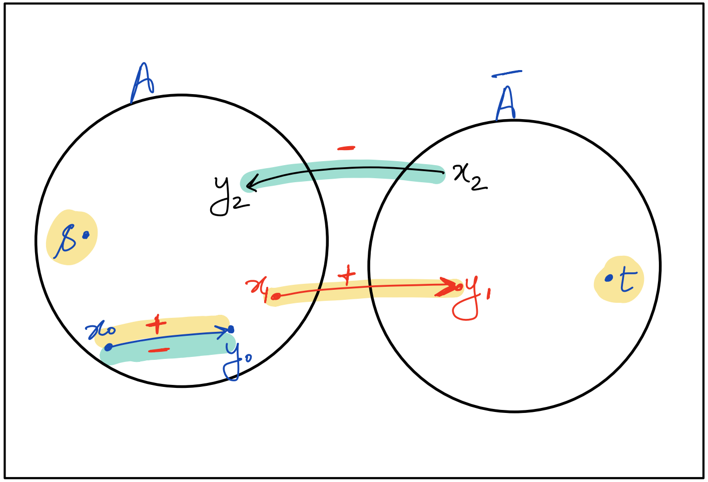
Property of Flows & Cuts

Property: For any (s, t)-cut $(A, \bar{A})$ and any flow f,

$$\text{value}(f) = f_{out}(A) - f_{in}(A)$$

Proof: For any (s, t)-cut $(A, \bar{A})$ and any flow f,

$$\begin{aligned}
\text{value}(f) &= f_{out}(s) \\
&= f_{out}(s) + \sum_{v \in A \setminus s} f_{out}(v) - f_{in}(v) \\
&= \sum_{v \in A} f_{out}(v) - f_{in}(v) \\
&= f_{out}(A) - f_{in}(A)
\end{aligned}$$



Property of Flows & Cuts

Property: For any (s,t)-cut $(A,\bar{A})$ and any flow f,

$$\text{value}(f) \;=\; f_{out}(A) - f_{in}(A)$$

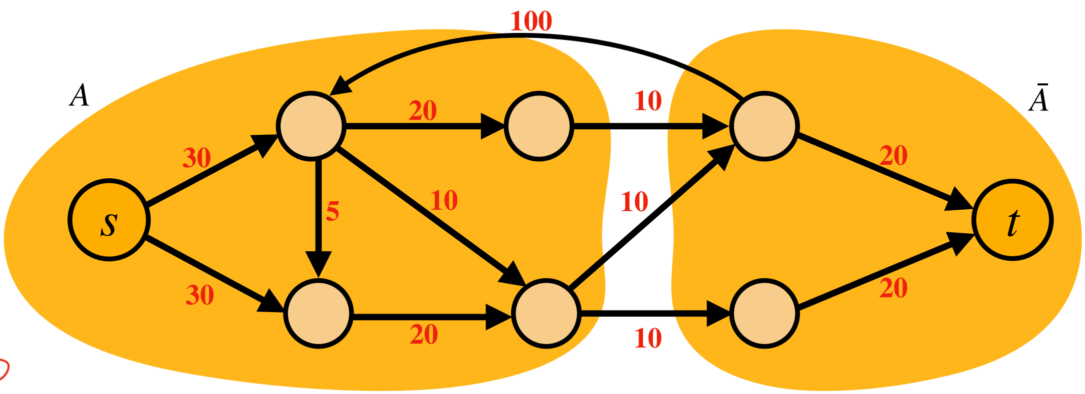
Corollary: For any (s,t)-cut $(A,\bar{A})$ and any flow f,

$$\text{value}(f) \;\leqslant\; c(A,\bar{A})$$

Implication: The value of (s,t)-max-flow is at most capacity of (s,t)-min-cut.

Eg.

(s,t)-max flow value = 30



$c(A,\bar{A}) = 30$

Max-flow Min-cut Theorem

Theorem: The value of (s, t)-max-flow is same as the capacity of (s, t)-min-cut.

Proof idea:

1. We showed the value of (s, t)-max-flow is at most the capacity of (s, t)-min-cut.

2. We will next show existence of a flow f and cut $(A, \bar{A})$ satisfying

$$\text{value}(f) = c(A, \bar{A}).$$

Max-Flow Algorithm

Ford-Fulkerson-algo(G, s, t):

1. Initialise $f = 0$
2. **While**($\exists\, s \to t$ path in G_f):

    2.1 Let P be an $s \to t$ path in G_f

    2.2 Let $c_{min} = \min\{c(e) \mid e \in P\}$

    2.3 **For each** $(x, y) \in P$:

    If (x, y) is forward edge : $f(x, y) = f(x, y) + c_{min}$

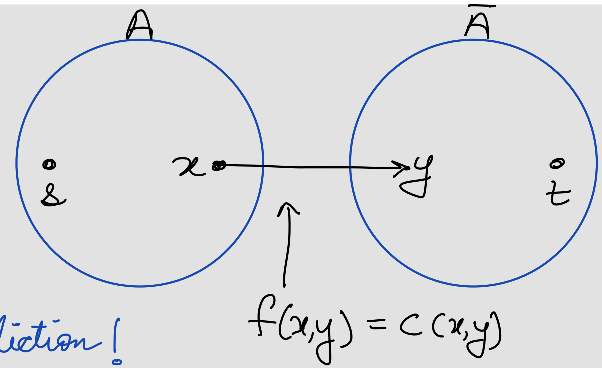
    If (x, y) is backward edge : $f(x, y) = f(x, y) - c_{min}$

3. Return f.

Proof: Let f be max-flow computed from Ford Fulkerson algorithm.

Let A = vertices reachable from s in G_f, and let $\bar{A} = V \backslash A$.

Claim 1: For each edge $(x, y) \in A \times \bar{A}$, $f(x, y) = c(x, y)$.

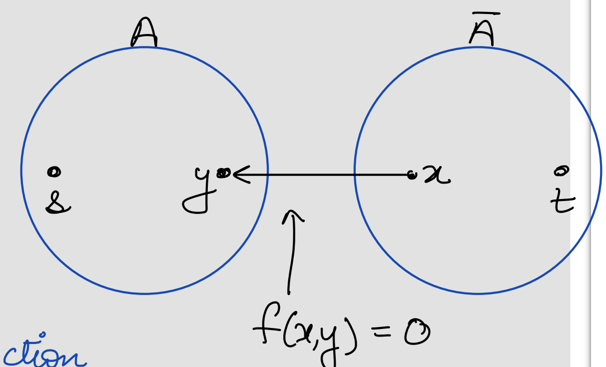


Suppose $f(x,y) \lneq c(x,y)$.

Then $c_r(x,y) > 0$

$\Rightarrow$ y is reachable from s in G_f. Contradiction!

Claim 2: For each edge $(x, y) \in A \times \bar{A}$, $f(x, y) = 0$.



Suppose $f(x,y) > 0$.

Then $c_r(y,x) > 0$.

$\Rightarrow$ x is reachable from y in G_f. Contradiction

Implication:

$$\left.\begin{array}{l} f_{out}(A) = c(A,\bar{A}) \\ f_{in}(A) = 0 \end{array}\right\} \Rightarrow \text{value}(f) = f_{out}(A) - f_{in}(A) = c(A,\bar{A})$$

Ford Fulkerson Algorithm

Theorem: The (s, t)-flow computed by Ford-Fulkerson is optimal.

Running time: 

You can assume that all capacities are at least 1.

SCENARIO

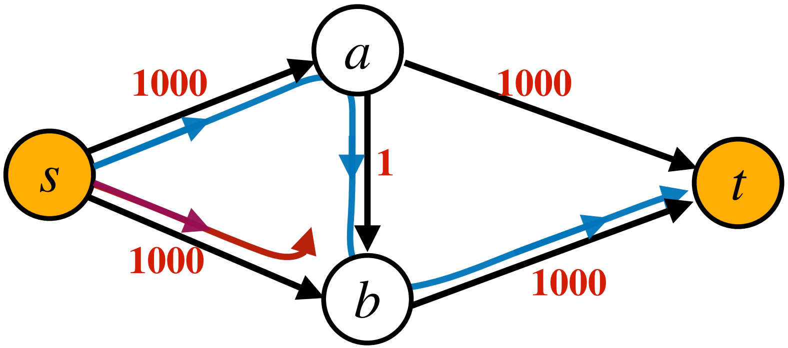
When all edge capacities are integers, and F = value of max-flow.

Then,

Time taken by Ford Fulkerson

$= O(F \cdot (m+n))$

How bad can be running time of Ford Fulkerson?



Remark:

The edge (a,b) disappears / reappears O(1000) times.

Claim:

Number of iterations can be exponential in input size

Max-Flow Efficient Algorithms

<u>Edmonds-Karp-algo(G, s, t):</u>

1. Initialise $f = 0$
2. **While**($\exists\ s \to t$ path in G_f):

    2.1 Let P be an $s \to t$ **shortest-path** in G_f

    2.2 Let $c_{min} = \min\{c(e) \mid e \in P\}$

    2.3 **For each** $(x, y) \in P$:

    If (x, y) is forward edge : $f(x, y) = f(x, y) + c_{min}$

    If (x, y) is backward edge : $f(x, y) = f(x, y) - c_{min}$

3. Return f.

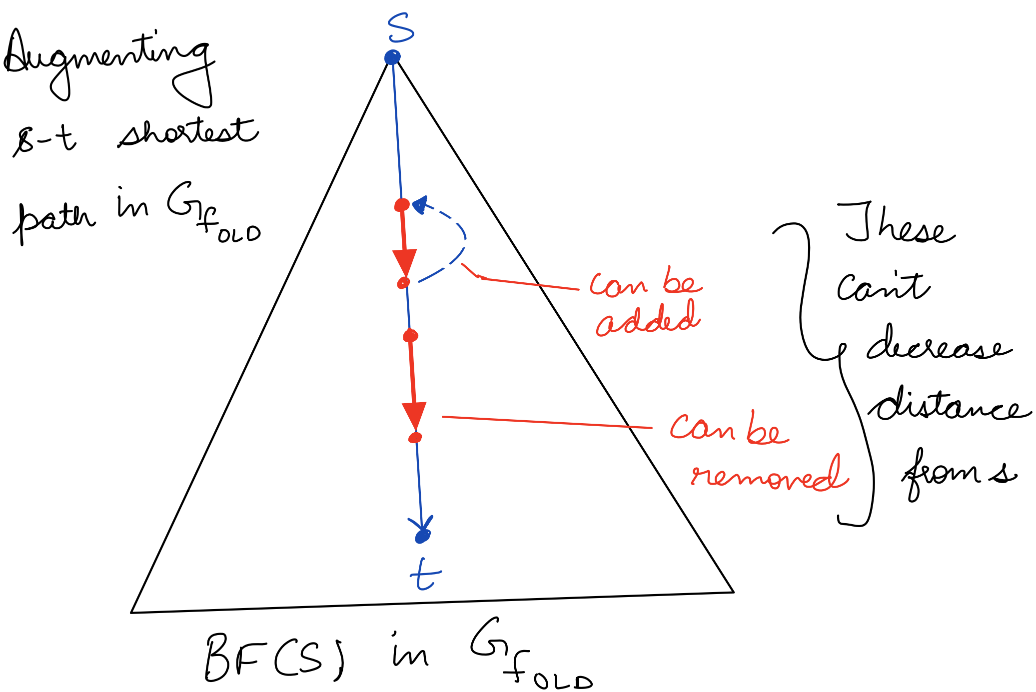
Claim:

Number of iterations is $O(mn)$

Thus, time $= O(m \cdot n \cdot (m+n))$

Observations

Claim 1: The distances of vertices from s in G_f can only increase with time.



When we move from $G_{f_{old}}$ to $G_{f_{new}}$ distance of vertices from 's' CANNOT decrease.