Maximum flow

To remember: max flow = min cut

Reading

- Dasgupta et al.: Chapter 7.2
- Cormen et al. : Chapter 26.1-2

Maximum Matching

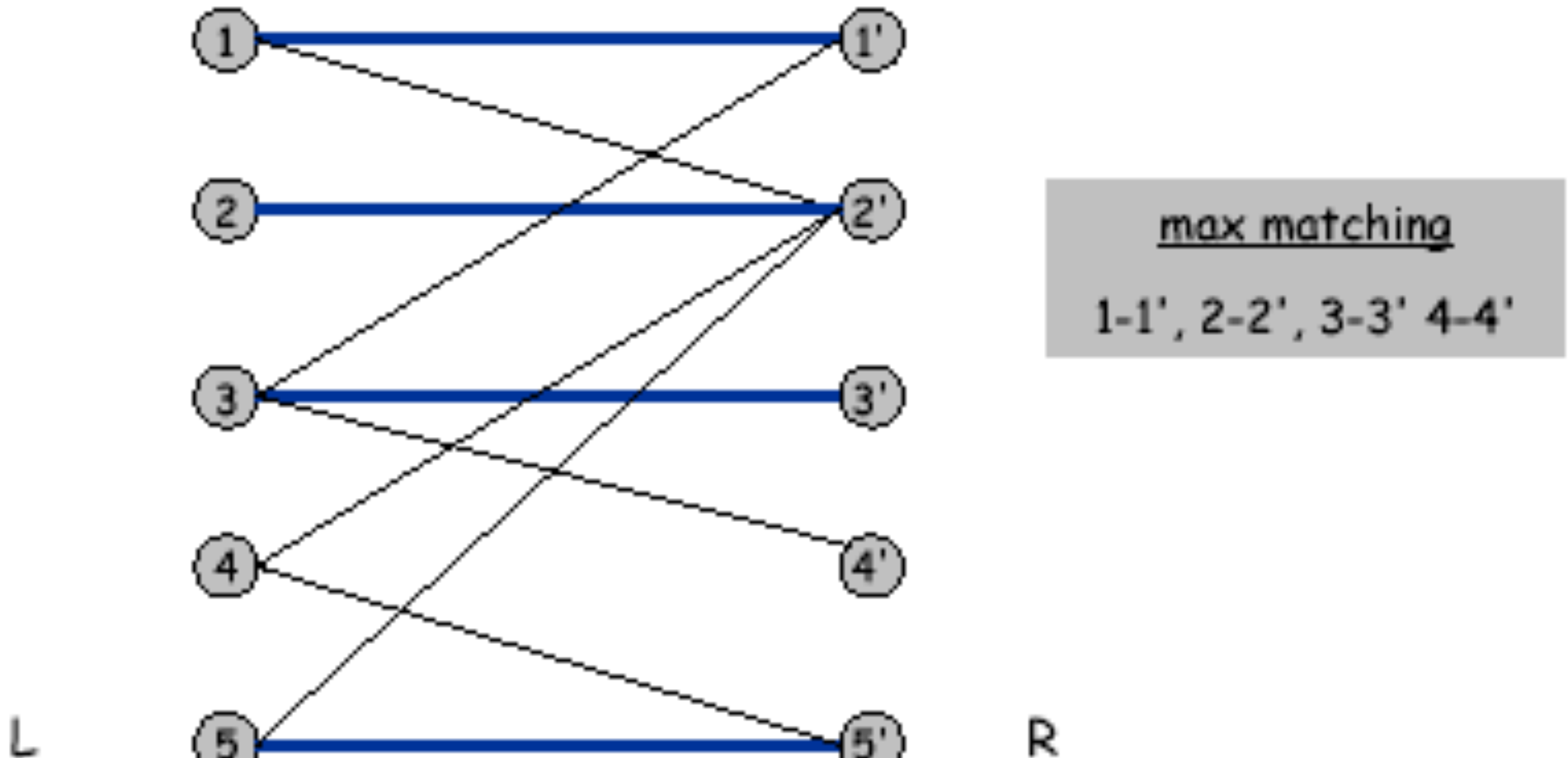
- A = HKU CS students
- B = HK pop singers
- On Apr 1, each singer will have dinner with a HKU CS student.
- Each student can specify up to 12 choices of singers.
- Find the maximum number of student-singer pairs.

Greedy algorithm ? Dynamic programming ?

Application of max flow: Bipartite matching

Input: undirected, bipartite graph G = (L, R, E).

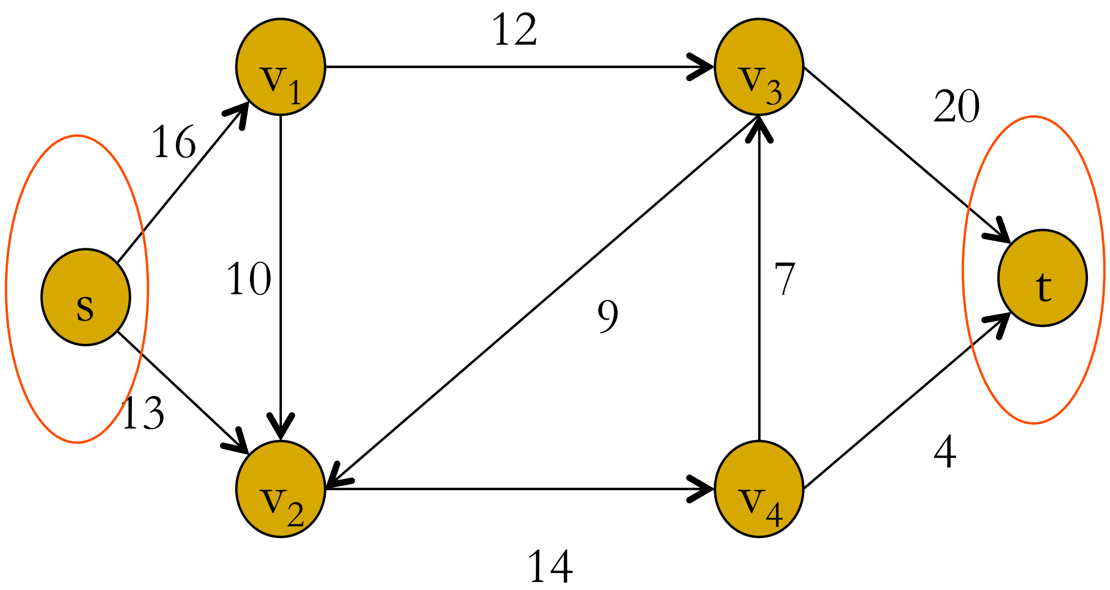
- M ⊆ E is a s matching if each node (vertex) appears in at most one edge in M.
- Maximum matching: find a matching with max number of edges.



Networks and Flow

- A flow network or simply network $G = (V, E, c)$ is a directed graph in which every edge (u,v) has a capacity $c(u,v) \geq 0$.

  G has two distinguished vertices: a source s and a sink t.
- Example:



Imagine an edge as a pipe.
Capacity = liter per min.

Flow

zero or positive

A flow f on G assigns a real value to every edge in G (i.e., $f\colon E \to R$) that satisfies two constraints:

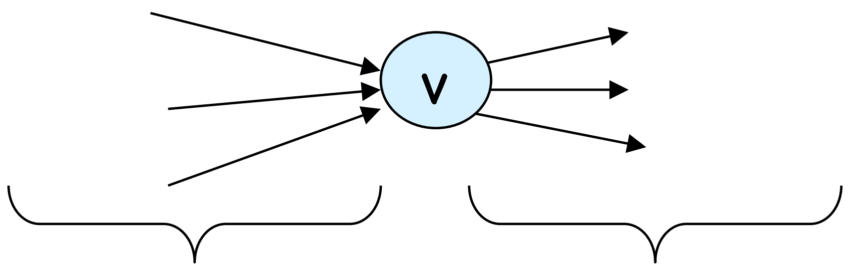
- Capacity constraint:

  For every edge $(u,v) \in E$, $f(u,v) \le c(u,v)$.

- Conservation constraint:

  For every vertex $v \in V - \{s, t\}$,

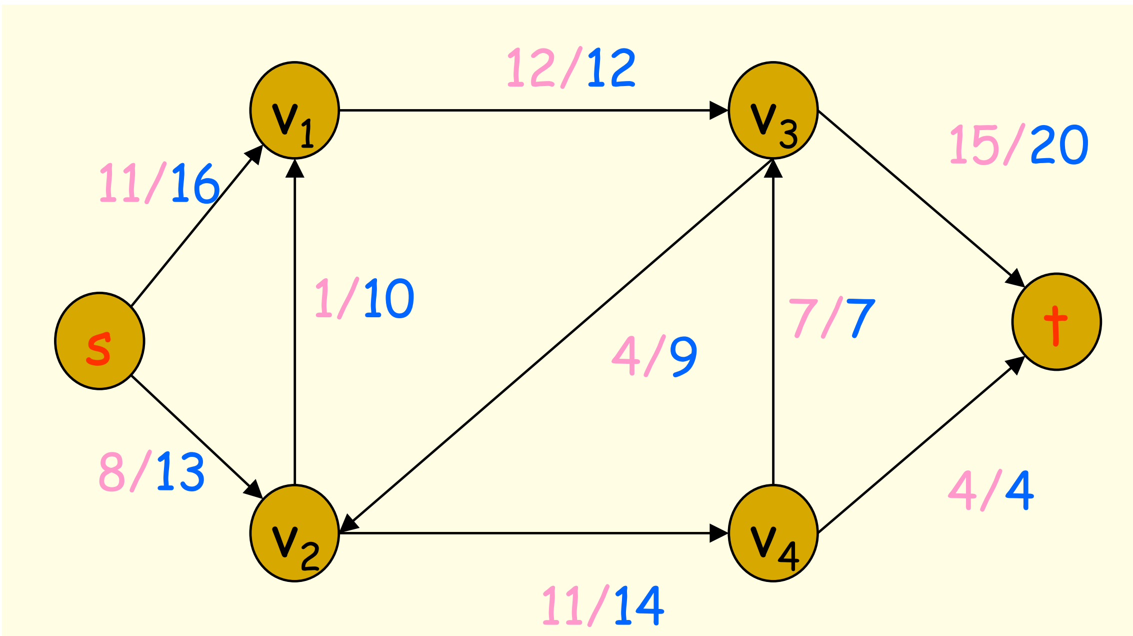
$$\sum_{(x,v)\in E} f(x,v) = \sum_{(v,y)\in E} f(v,y).$$



total inflow = total outflow

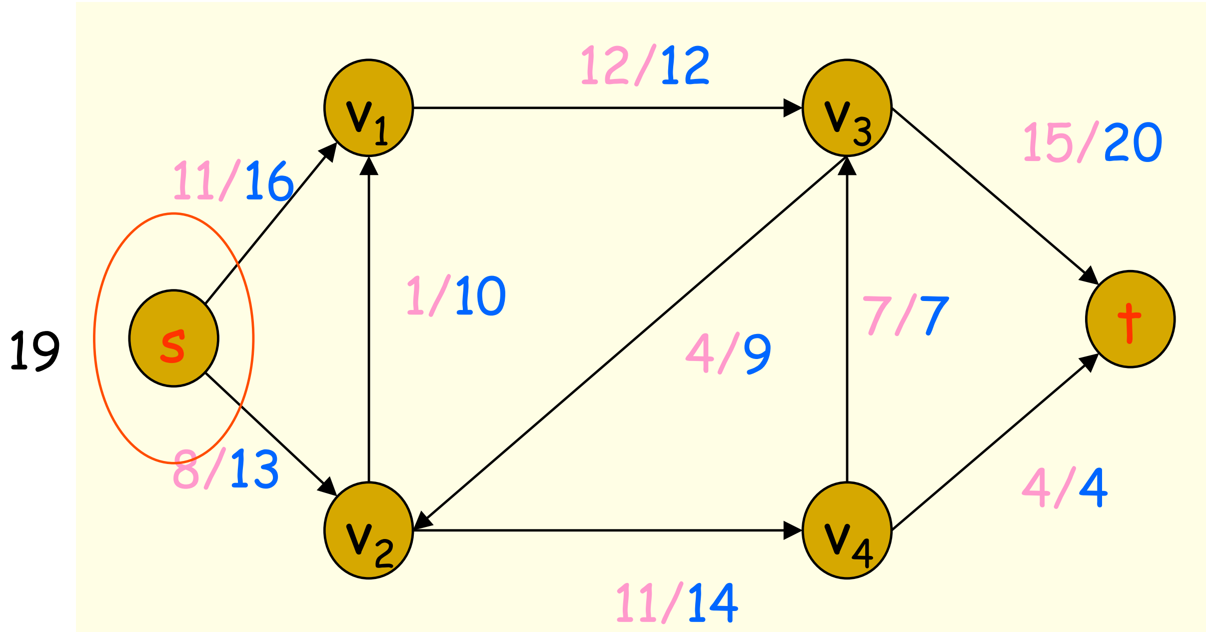
Example

- Capacity constraint:
  For every edge $(u,v) \in E$, $f(u,v) \leq c(u,v)$.
- Flow conservation constraint:
  For every vertex $v \in V - \{s, t\}$, $\sum_{(x,v)\in E} f(x,v) = \sum_{(v,y)\in E} f(v,y)$



Value (or size) of a flow f

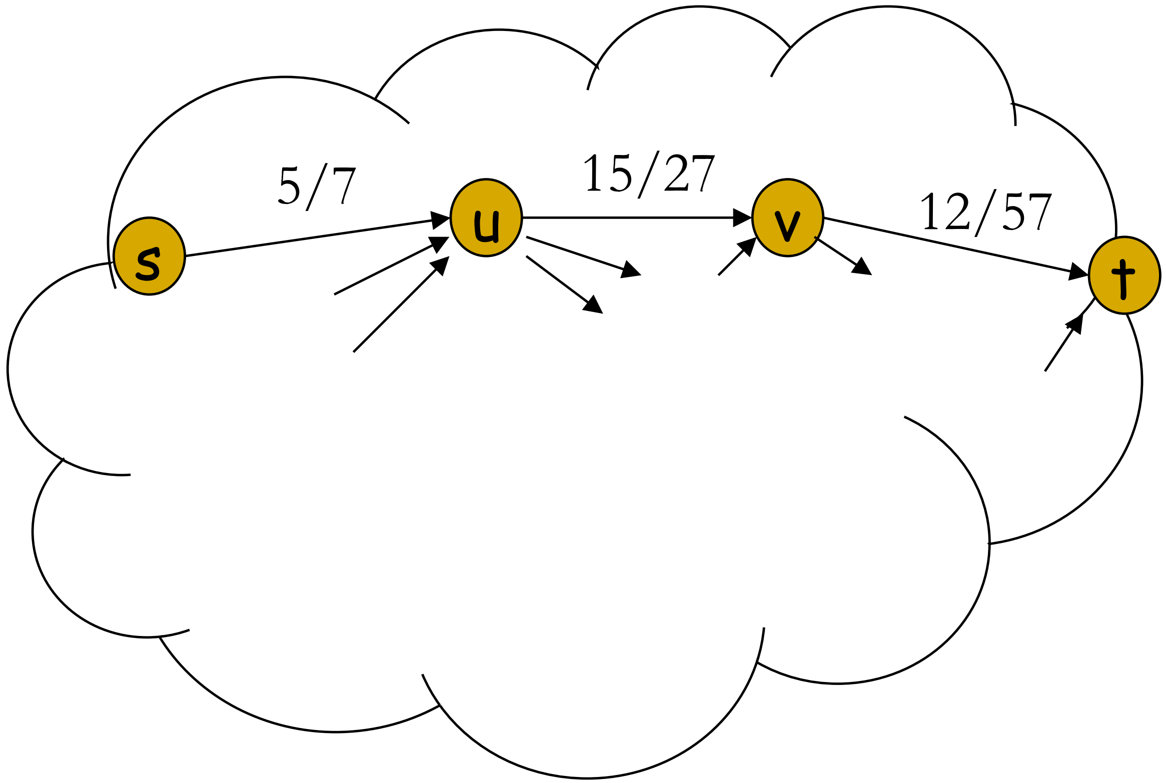
- **Notations:** value(f), size(f), v(f)
- value(f) = $\sum_{(s,y)\in E} f(s,y) = \sum_{(v,t)\in E} f(v,t)$.
  - Example: value(f) = 19



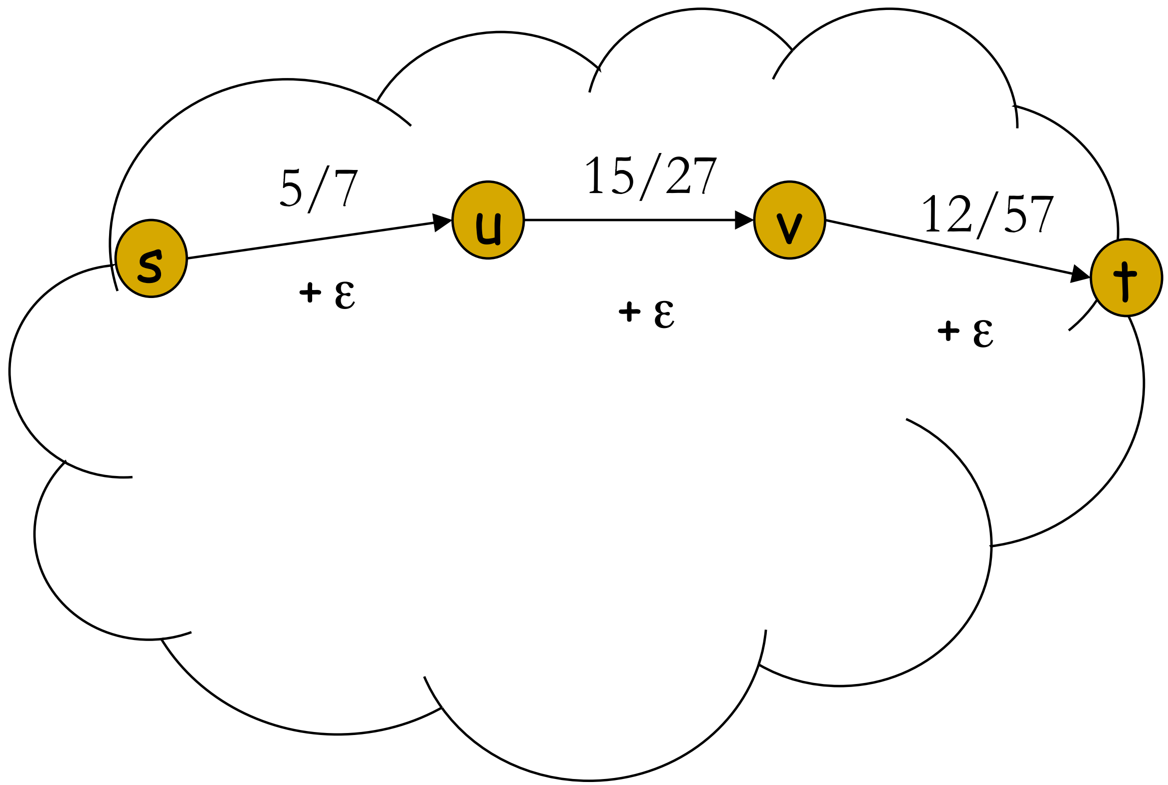
Maximum flow

- Given G=(V, E, c), find a flow of G with the maximum value.
- How to solve this problem?
- Greedy algorithm:
  - Start with some trivial flow of G, say, an empty flow.
  - Repeatedly augment the current flow until the flow is maximum.
- Questions:
  - How to systematically augment a flow of G.
  - How to determine whether the current flow is maximum.

A path from s to t.

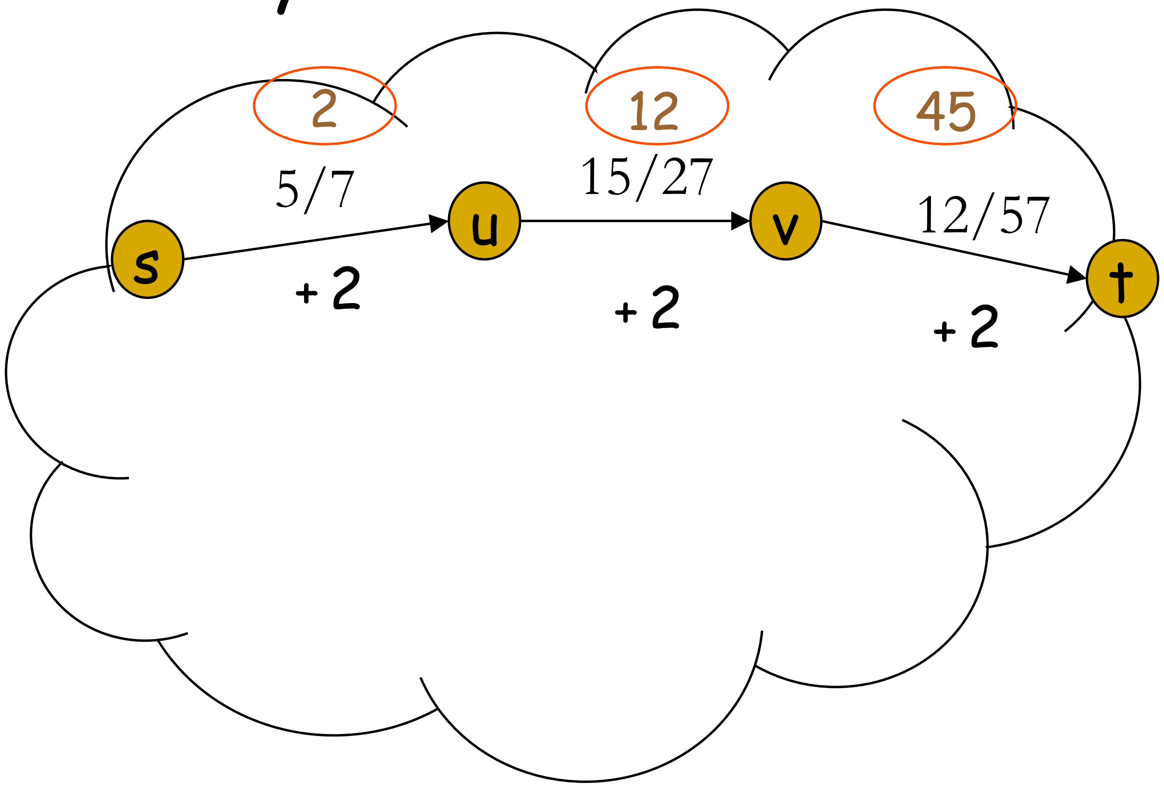


Push a small increase



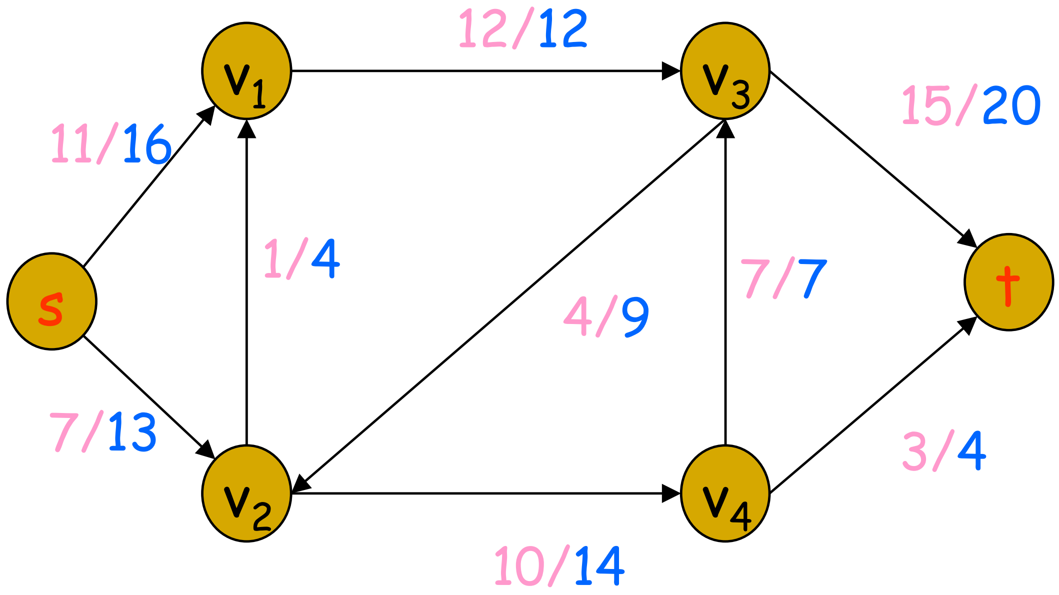
How much to push? As much as possible

Note that ε cannot be larger than the smallest residual capacity on the path.
I.e., at most 2 in the following example. Then the flow increases by 2.

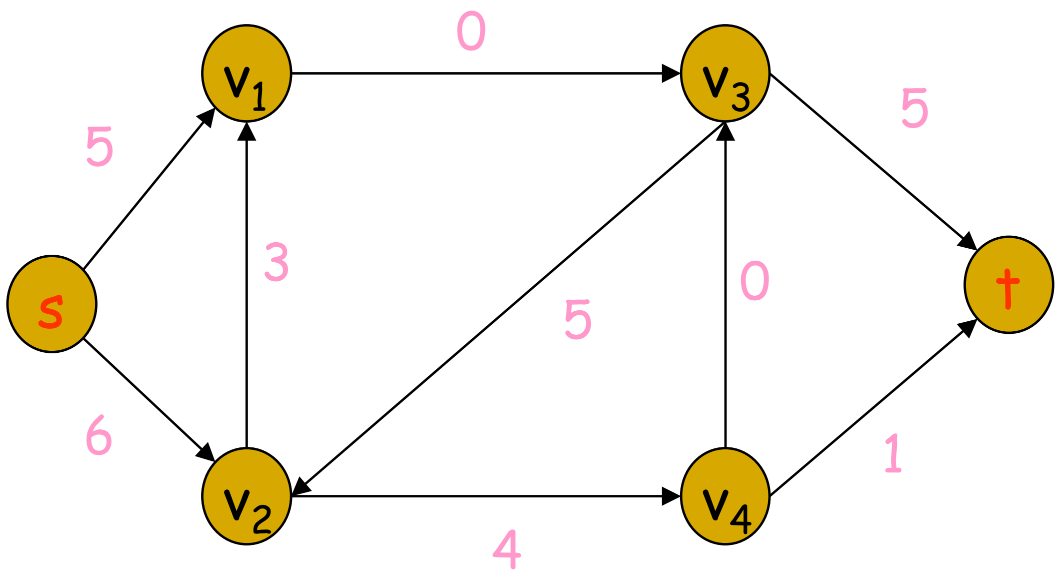


A systematic method for augmenting a flow

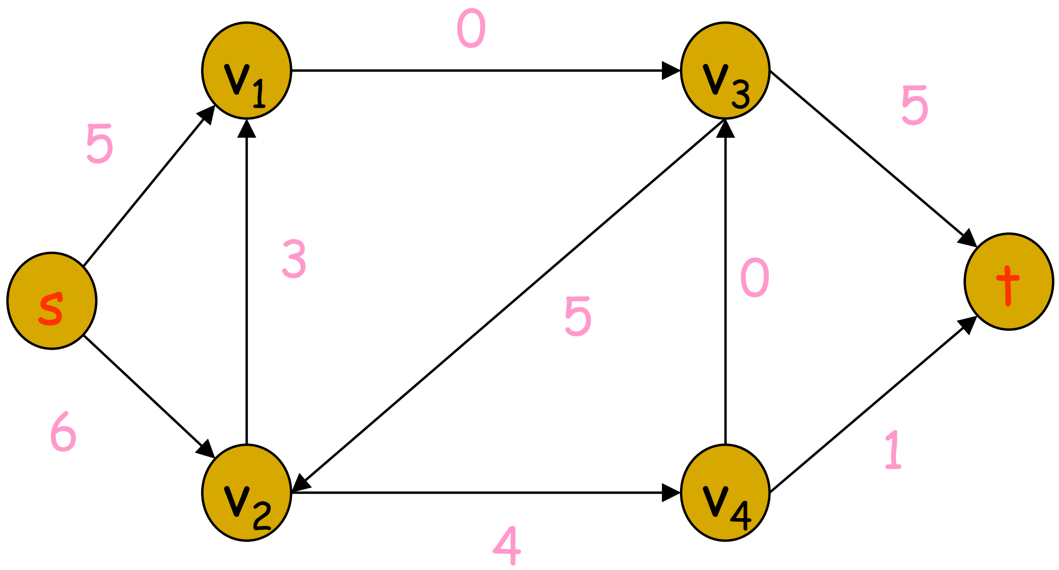
A network with an initial flow



Step 1: For every edge $(u,v) \in E$, calculate the residual capacity = $c(u,v)-f(u,v)$

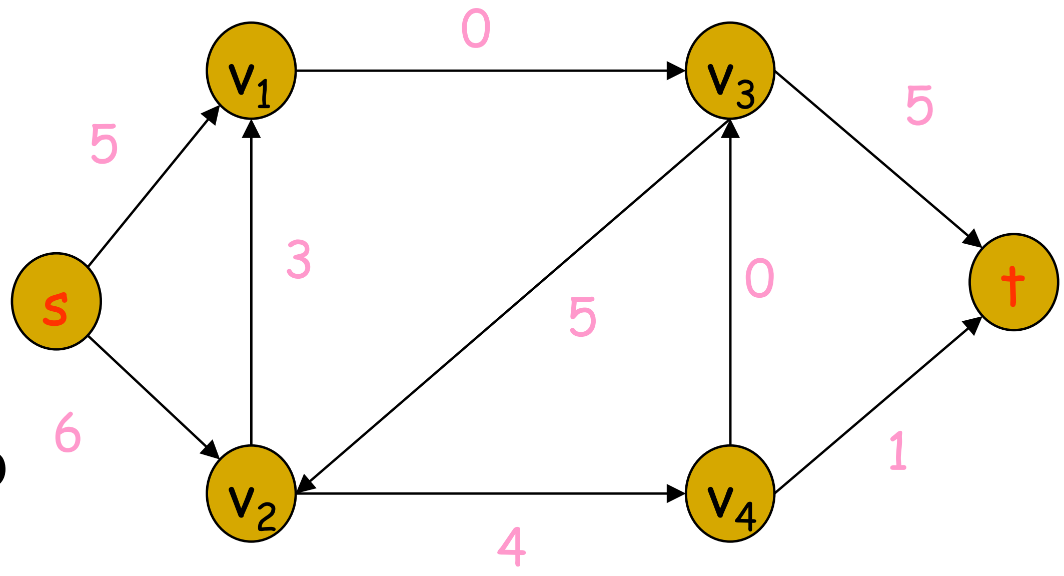


Find the augmenting path



G_f

Residual Network

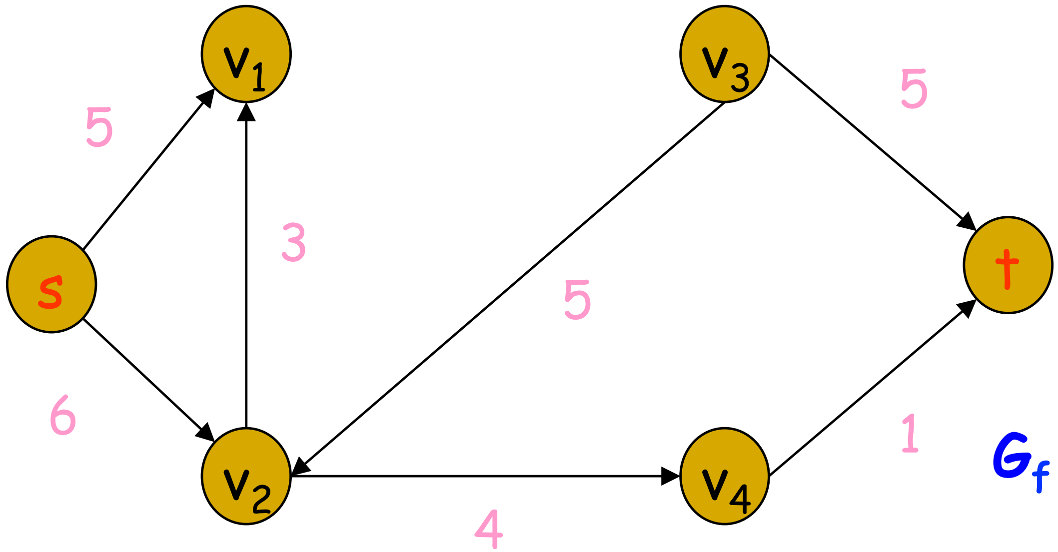


Step 2: remove edges with zero residual capacity

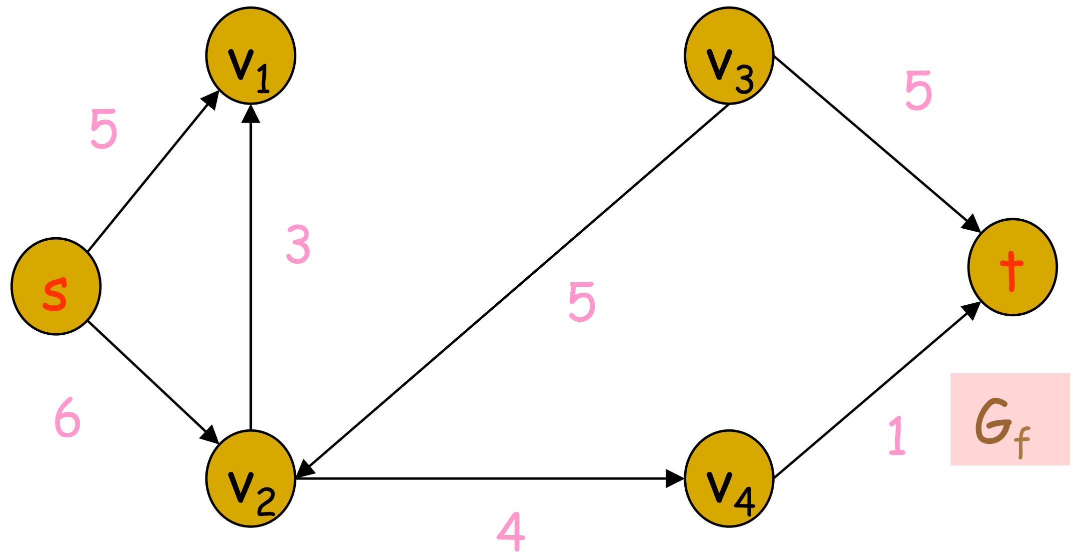
(useless for augmentation).

Let the resulting graph after Steps 1 and 2 be G_f.

Residual Network



Find the augmenting path

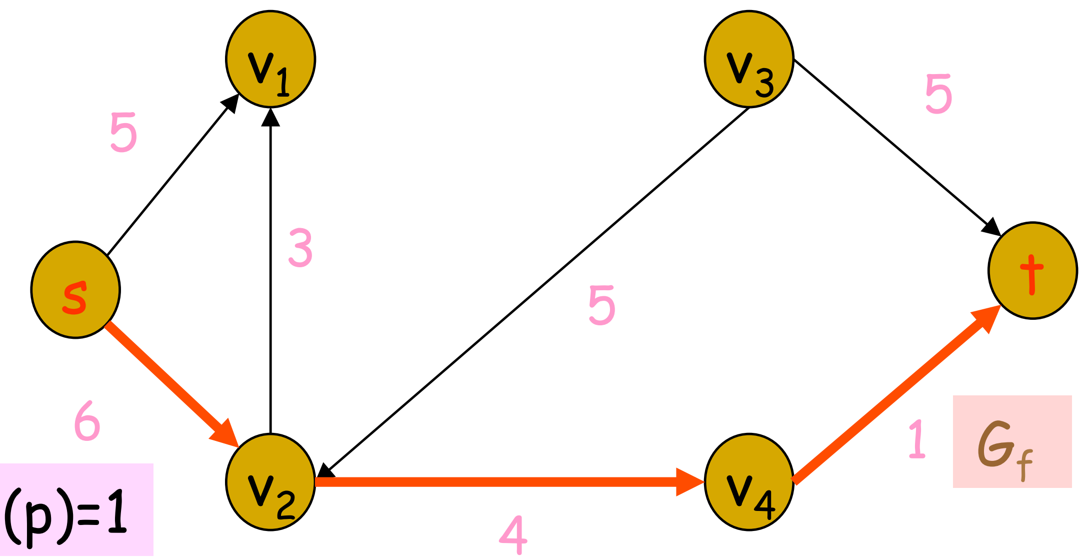


Augmenting path

Step 3: Find any path p from s to t in G_f.

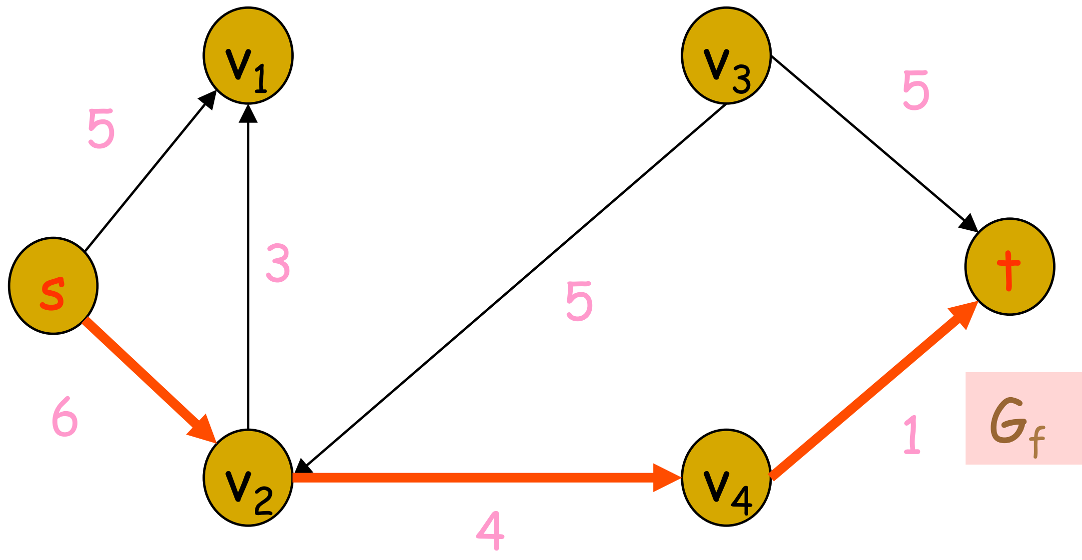
Let $rc(p) = \min_{(u,v)\in p}\{c(u,v) - f(u,v)\}$.

Note that $rc(p) > 0$.



rc(p)=1

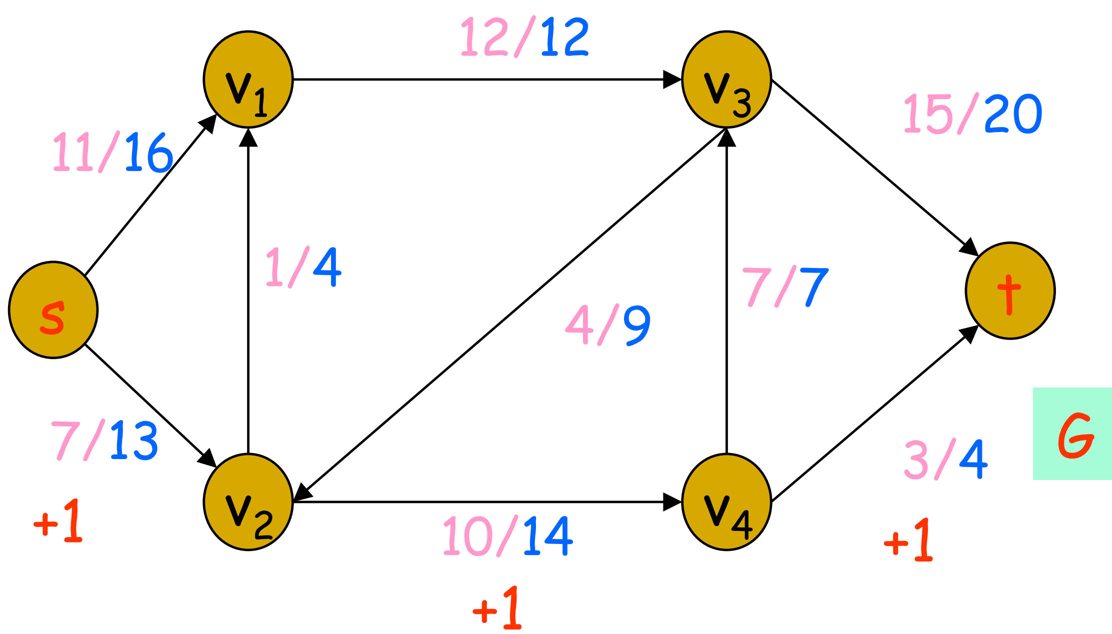
Update the flow



Step 4: For every edge e in p, increase its flow value by rc(p).

This results in a bigger flow f'.

$$f'(u,v) = \begin{cases} f(u,v) + rc(p) & \text{if } (u,v) \in p \\ f(u,v) & \text{otherwise} \end{cases}$$



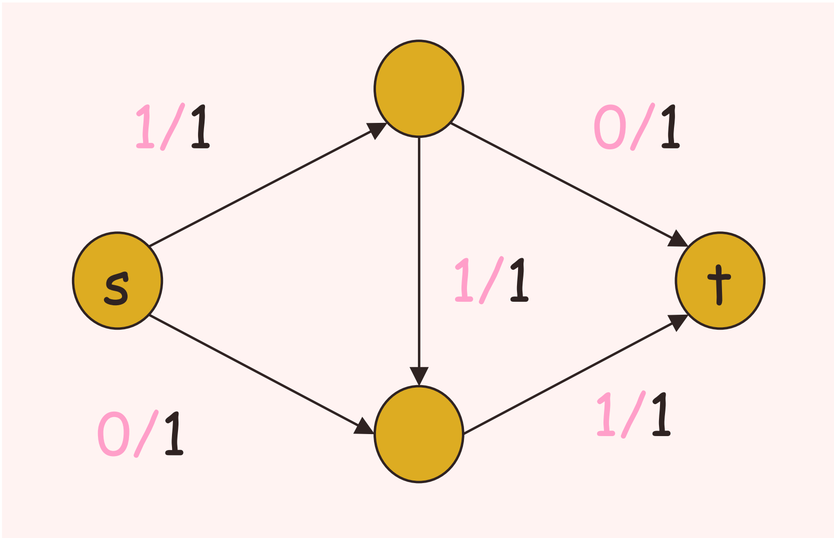
Summary

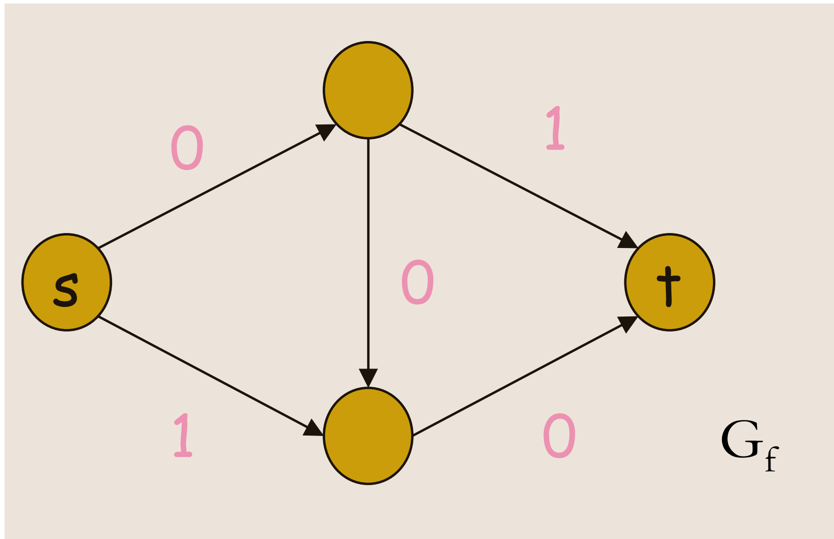
- Start with an initial flow.
- Repeat
  - Compute the residual capacity & residual network
  - Find an augmenting path
  - Update the flow

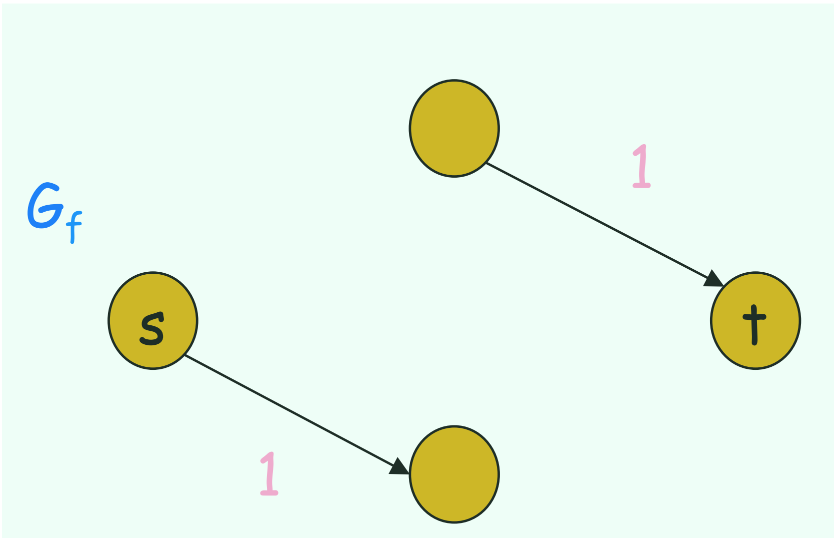
until no augmenting path is found (and no increase to the flow)

Can such a approach find the maximum flow?

- Not always.



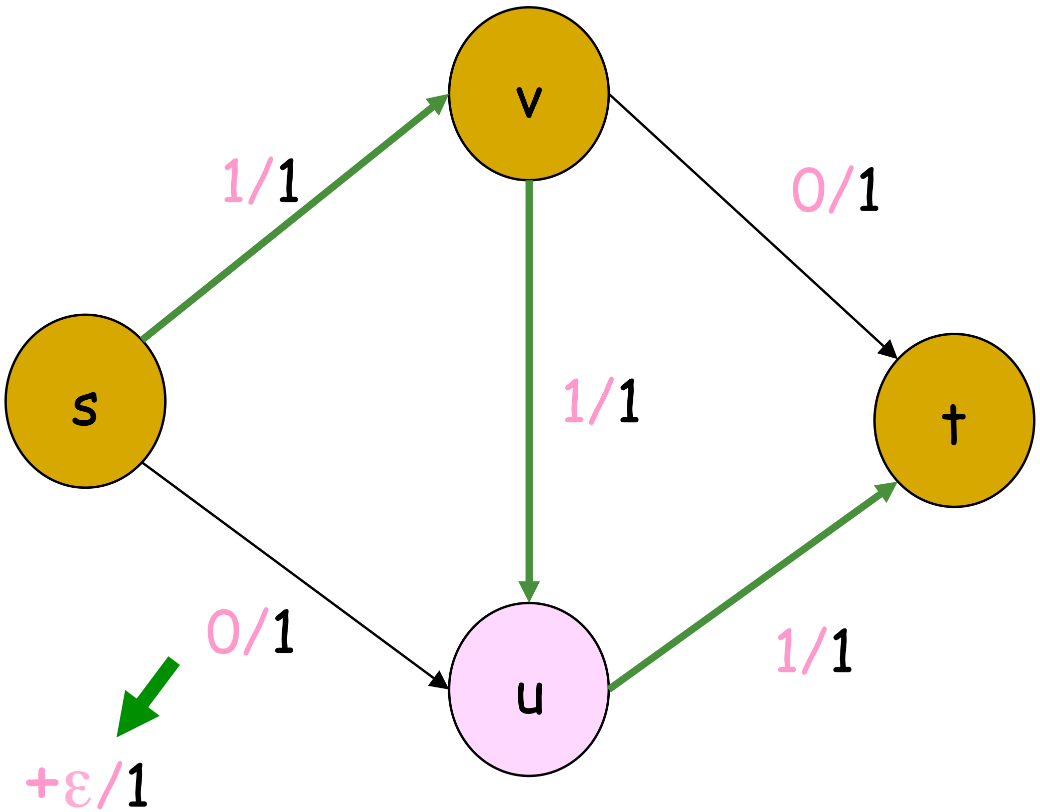




No augmenting path from s to t.

Ford-Fulkerson method for flow augmentation

Recall that we were stuck with this example.

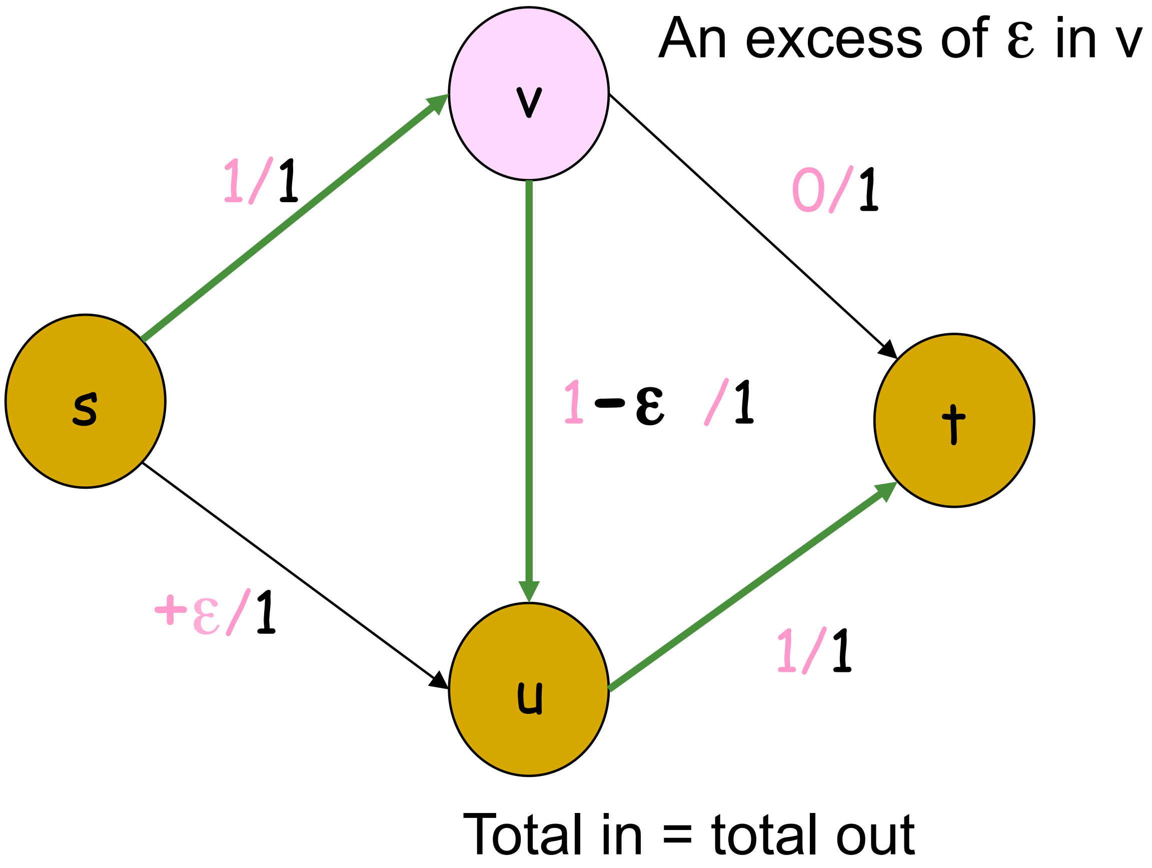


No way to push the excess away from u.

Reverse the flow

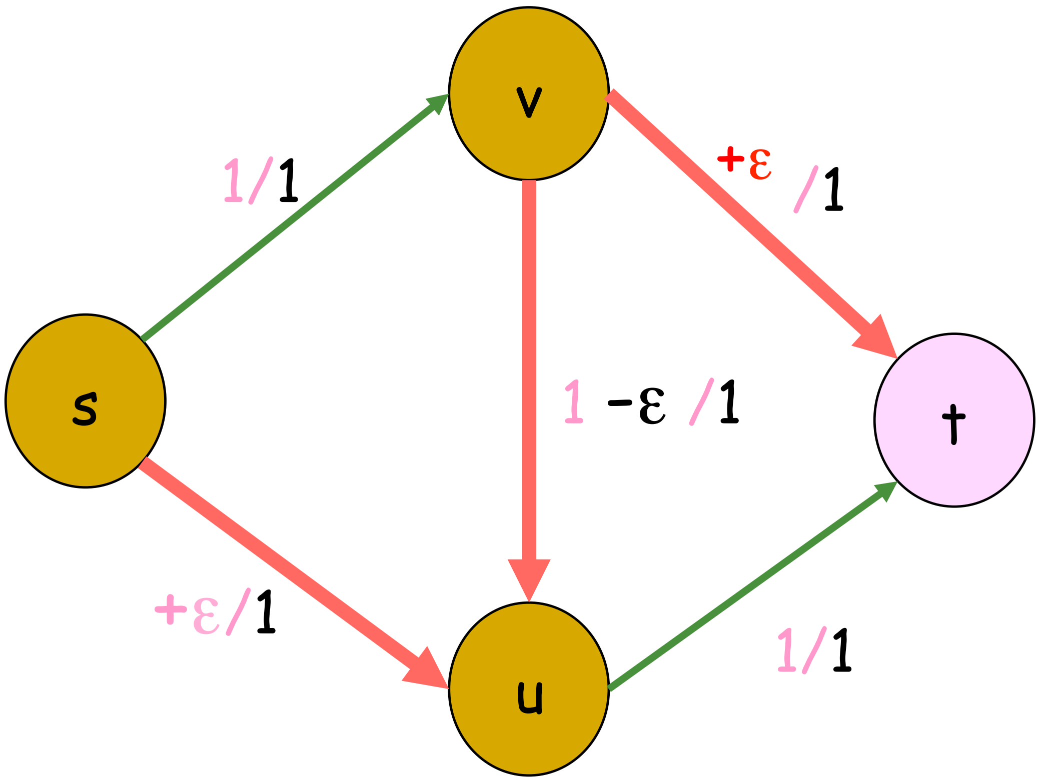
To balance the total inflow and total outflow, we can

- push away ε units from the node, or
- decrease the incoming flow to the node by ε units



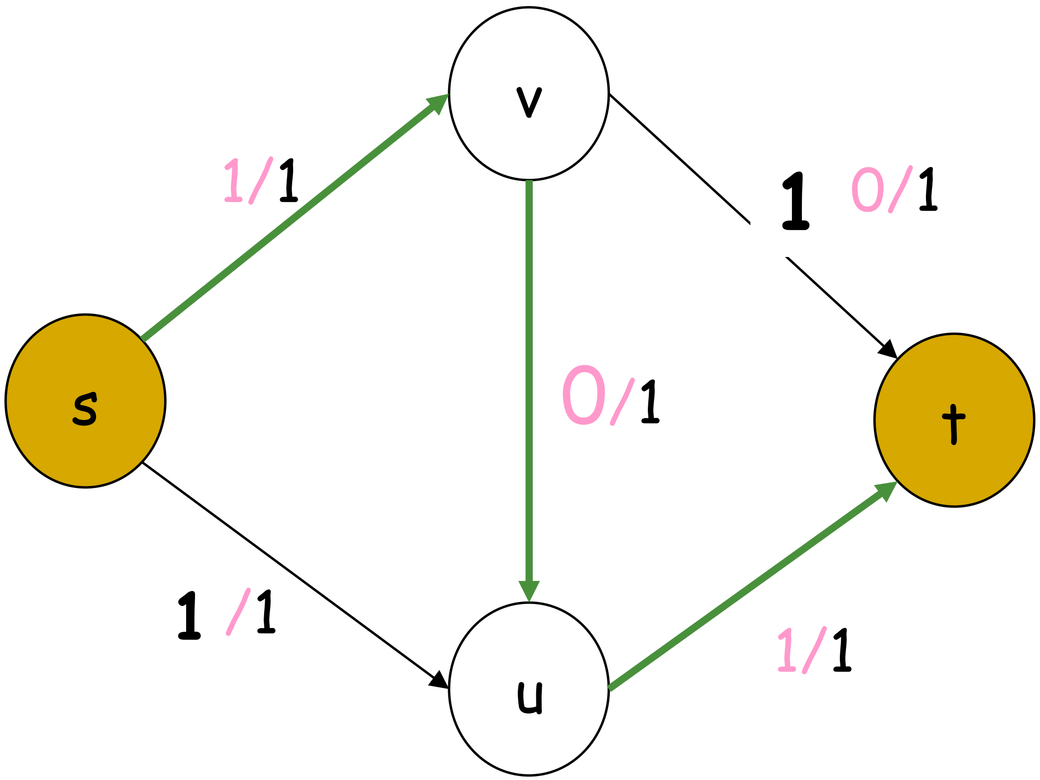
Augmentation path does exist

We can push an extra of ε units from s to t.



Maximum push along the augmenting path

- Push ε = 1 units of flow.



The Ford-Fulkerson method

Residual network G_f

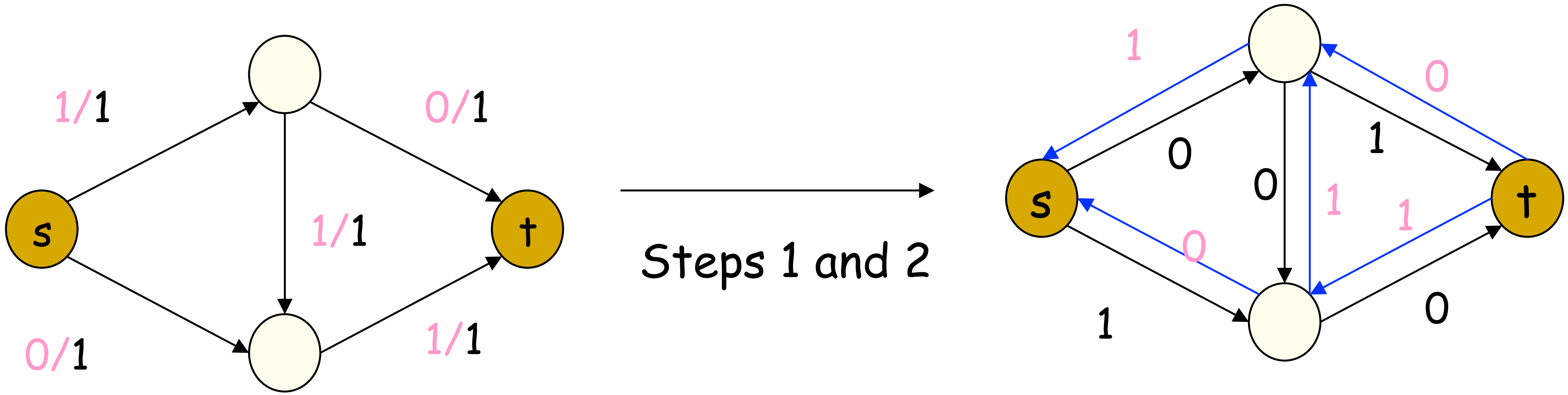
Step 1: for every edge $(u,v) \in E$,

add (u,v) (forward edge) to G_f with residual capacity = $c(u,v)$ - $f(u,v)$.

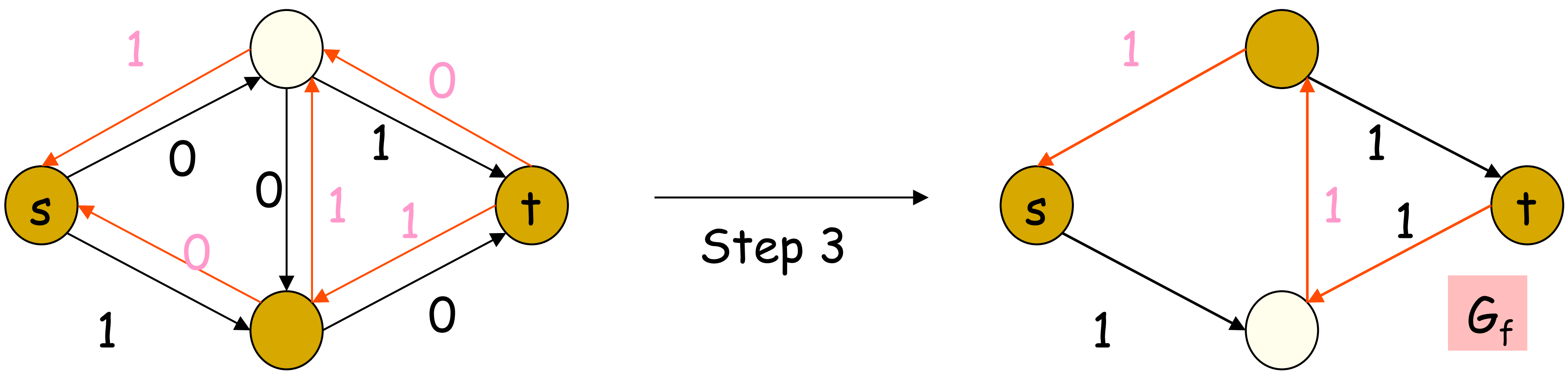
Step 2: for every edge $(u,v) \in E$,

add (v,u) (backward edge) to G_f with residual capacity = $f(u,v)$.

Step 3: remove all edges with 0 residual capacity.



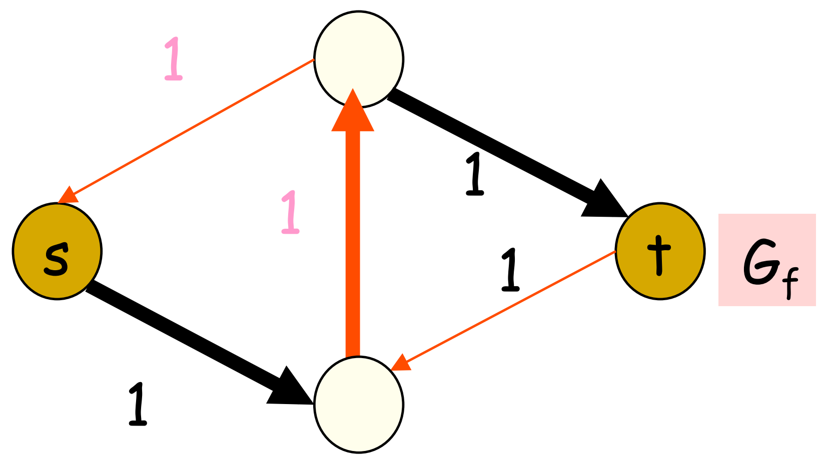
Augmenting path

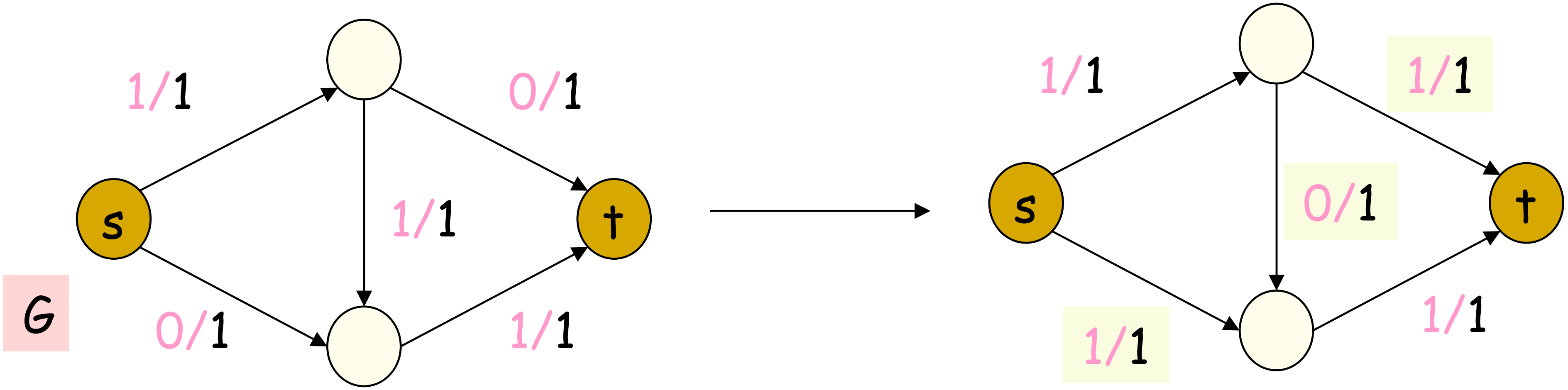


1. Find a path p in G_f , and compute the residual capacity rc of this path.
2. Augment G along p as follows:

$$f'(u,v) = \begin{cases} f(u,v) + rc & \text{if } (u,v) \in p \text{ and is a forward edge} \\ f(u,v) - rc & \text{if } (v,u) \in p \text{ and is a backward edge} \\ f(u,v) & \text{otherwise} \end{cases}$$

Example





Summary: Finding maximum flow

Start with zero flow f (i.e., f(u,v)=0 for all $(u,v) \in E$).

Repeat

- Construct the residual network G_f with forward & backward edges.
- Find a path p with residual capacity rc(p) from s to t in G_f
- If no such path exists, return f as the maximum flow.
- Otherwise, augment the flow f with respect to p.

- Sample run of the Ford and Fulkerson algorithm.

Correctness

???

Running time: O(nmC)

Assumption. All capacities are integers between 1 and C.

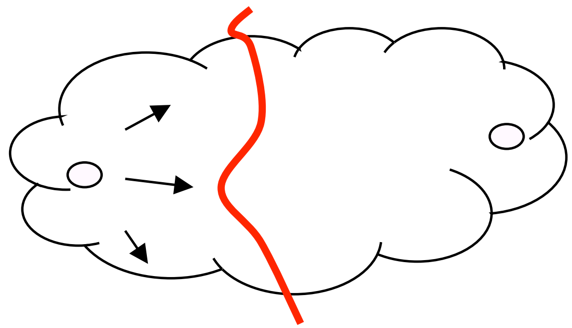
Lemma. The algorithm requires at most nC iterations (augmentations).

Proof. Each augmenting path increases the flow value by at least one. Maximum flow is bounded by nC.

Each iteration takes $O(n+m)$ time.

Correctness

Theorem. Let f^* be a flow such that G_{f^*} has no augmentation paths. Then f^* is a maximum flow.



Proof framework:

- f^* admits no augmenting paths

$\Rightarrow$ The network has a "cut" of with capacity = value(f^*).

- For all possible flow f', value(f') ≤ the capacity of the "cut" = value(f^*).

- Therefore, f^* admits no augmenting paths

$\Rightarrow$ value(f^*) is maximum.

Organization

Definition of a cut (slides 35-36)

Flow value lemma (slides 37-43)

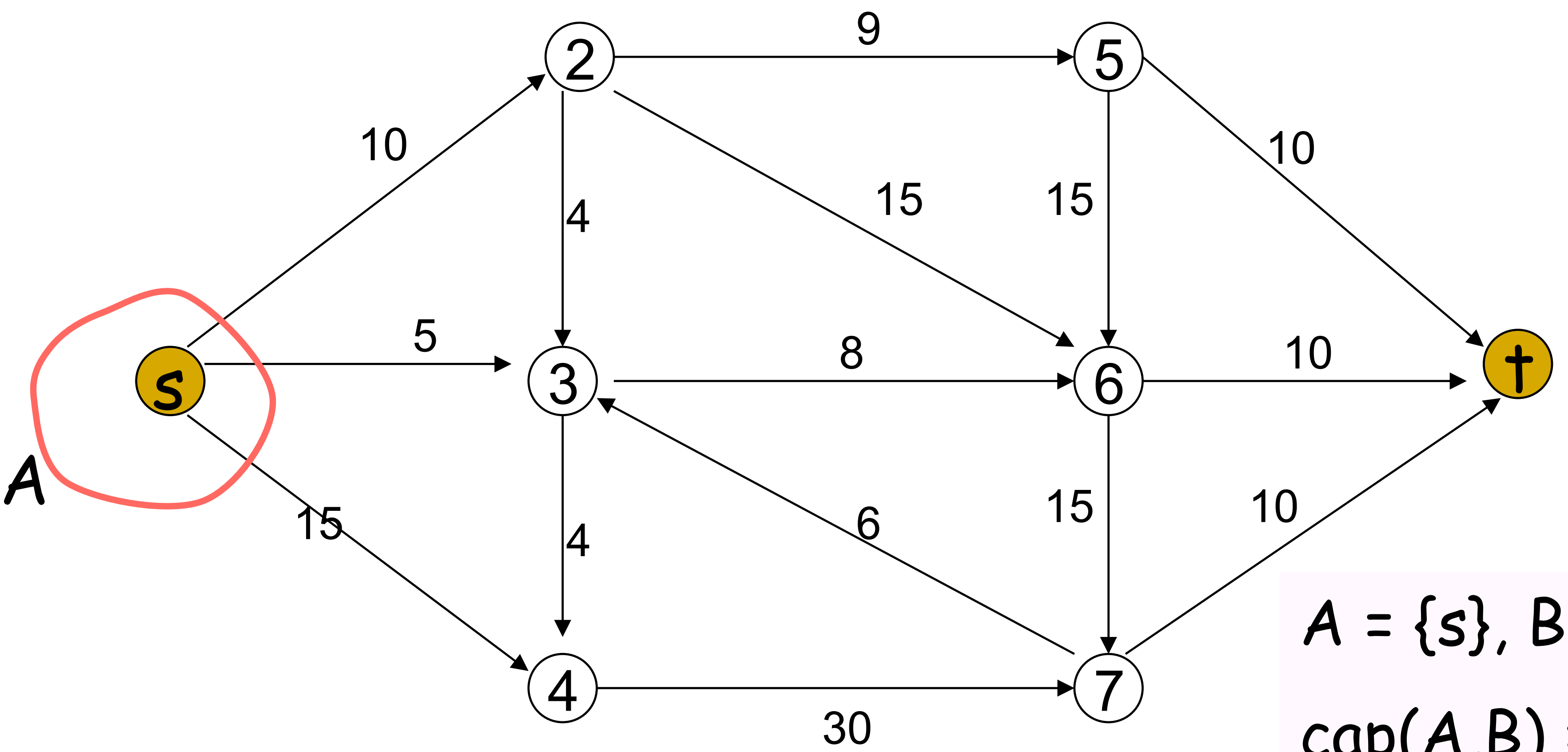
⇒ For all possible flow f', value(f') ≤ the capacity of a "cut".

⇒ If f* has no augmenting paths, then the network has a "cut" of with capacity = value(f*).

Cut

Definition.

- A s-t cut of G is a partition (A,B) of the vertices such that s in A and t in B.
- The capacity of a cut (A,B), cap(A,B), = $\sum_{e \text{ out of } A} c(e)$.

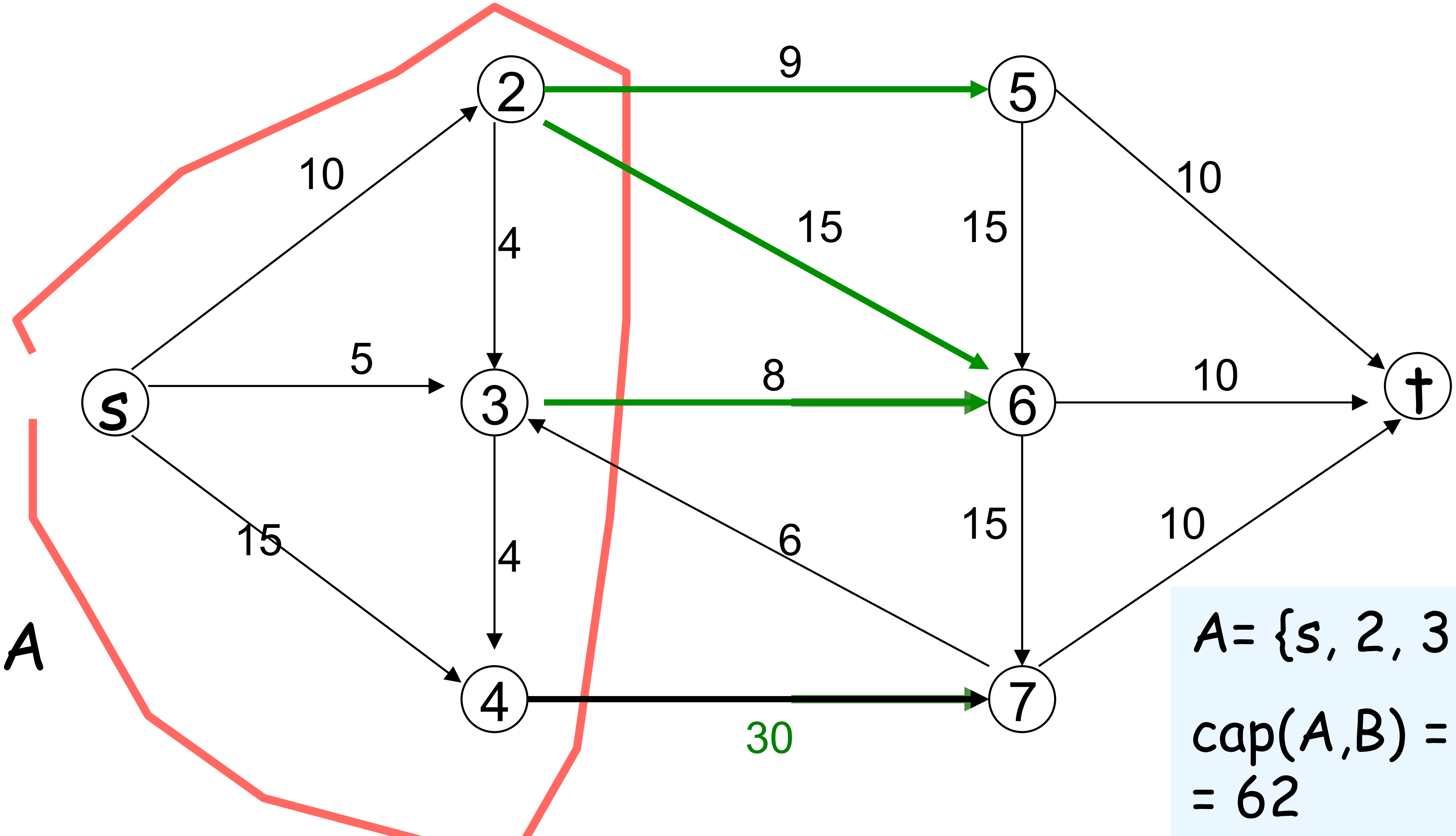


A = {s}, B = V – {s}.

cap(A,B) = 10+5+15 = 30

Cut

Definition. $cap(A,B) = \sum_{e \text{ out of } A} c(e)$.

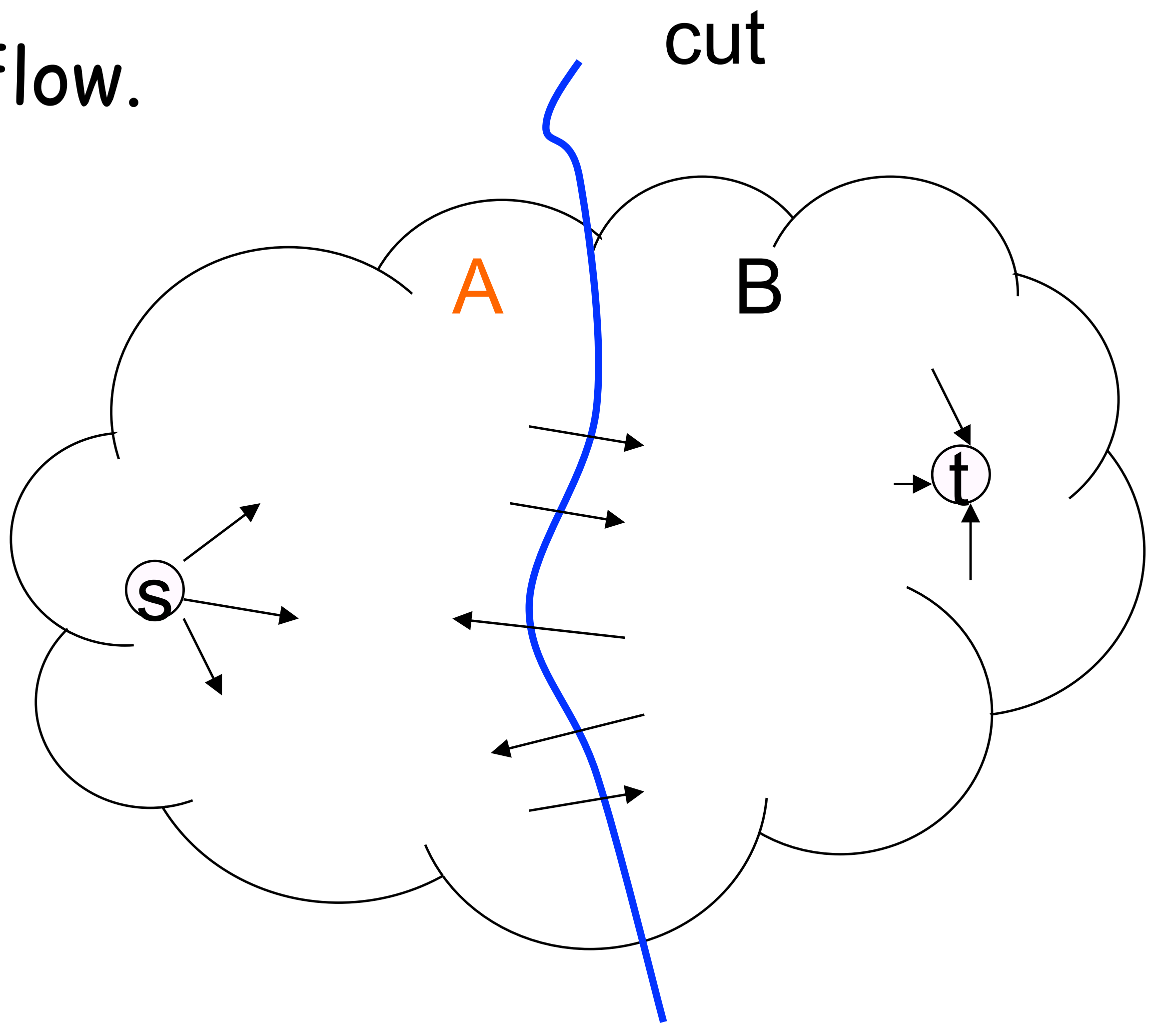


A= {s, 2, 3, 4}, B=V- A.

cap(A,B) = 9+15+8+30 = 62

Flow & Cut

Let f be a flow.

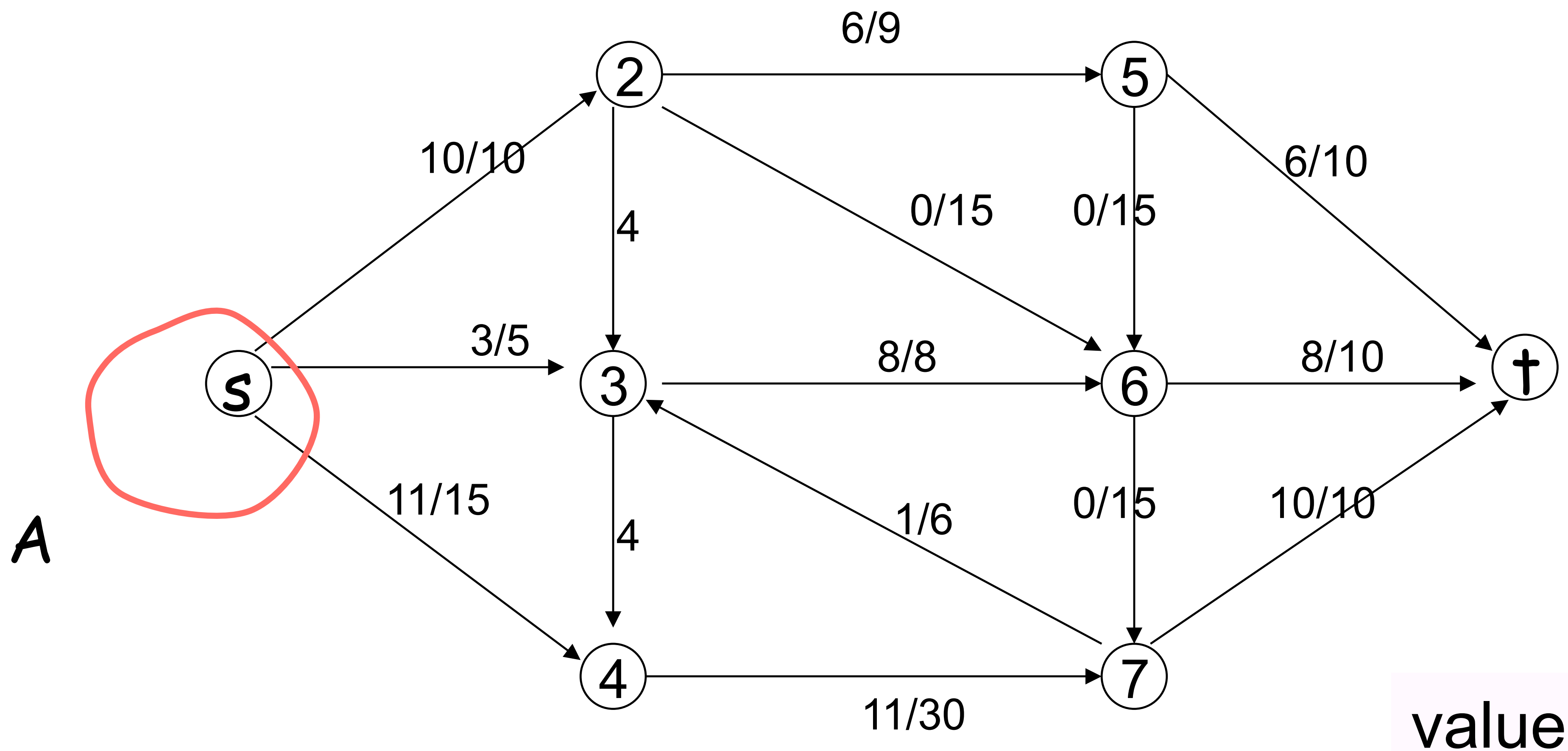


By definition, value(f) = total flow out of s.

Intuitively, value(f) = the net flow across the cut = flow from A to B minus flow from B to A.

Flow value lemma

Lemma. Let f be any flow of G, and let (A,B) be any s-t cut of G. Then value(f) = the net flow across the cut, i.e., $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$.

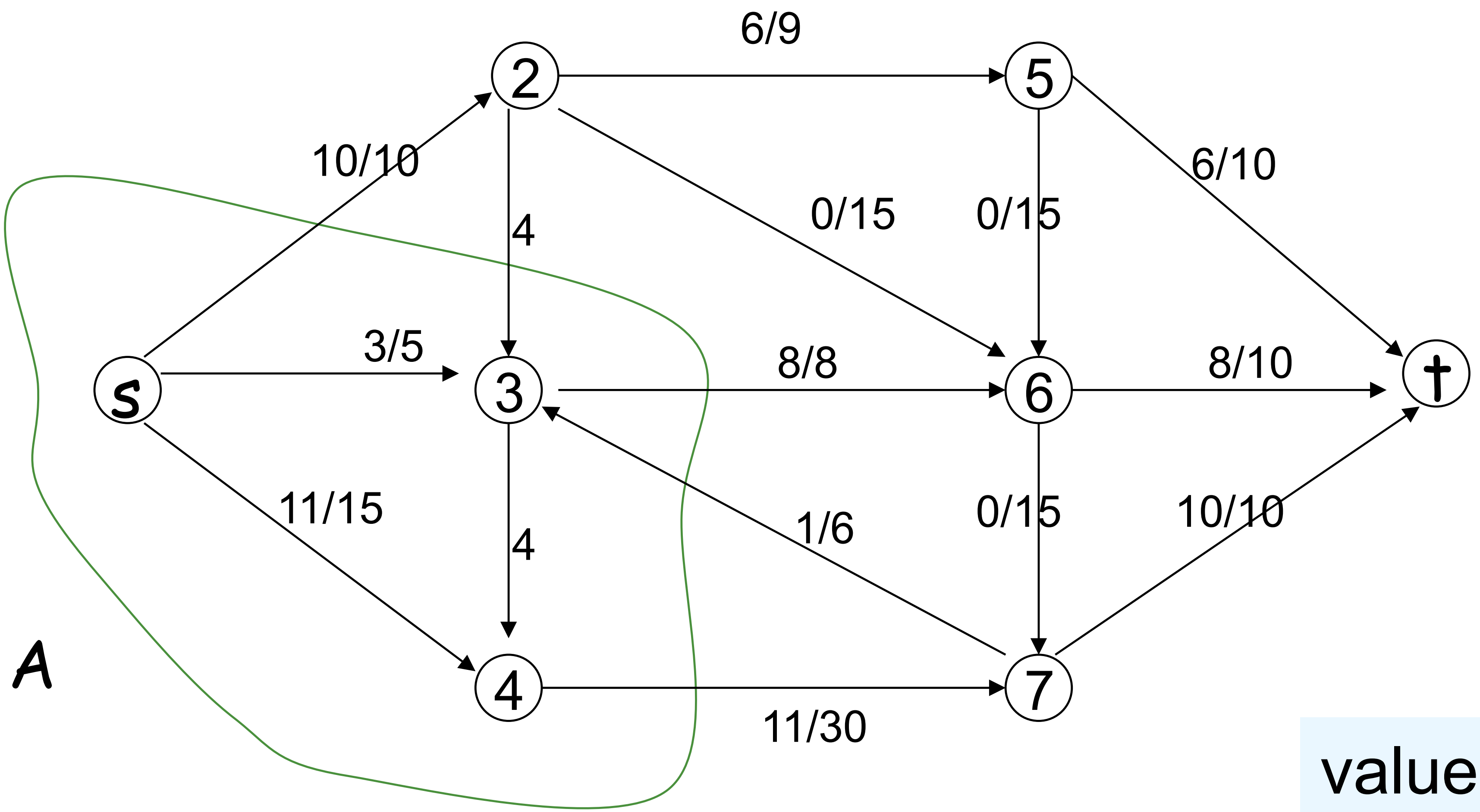


value(f) = 24

Another example

Let f be any flow of G, and let (A,B) be any s-t cut of G. Then value(f) = $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$.

Net flow = 11+8+10-4-1 = 24.

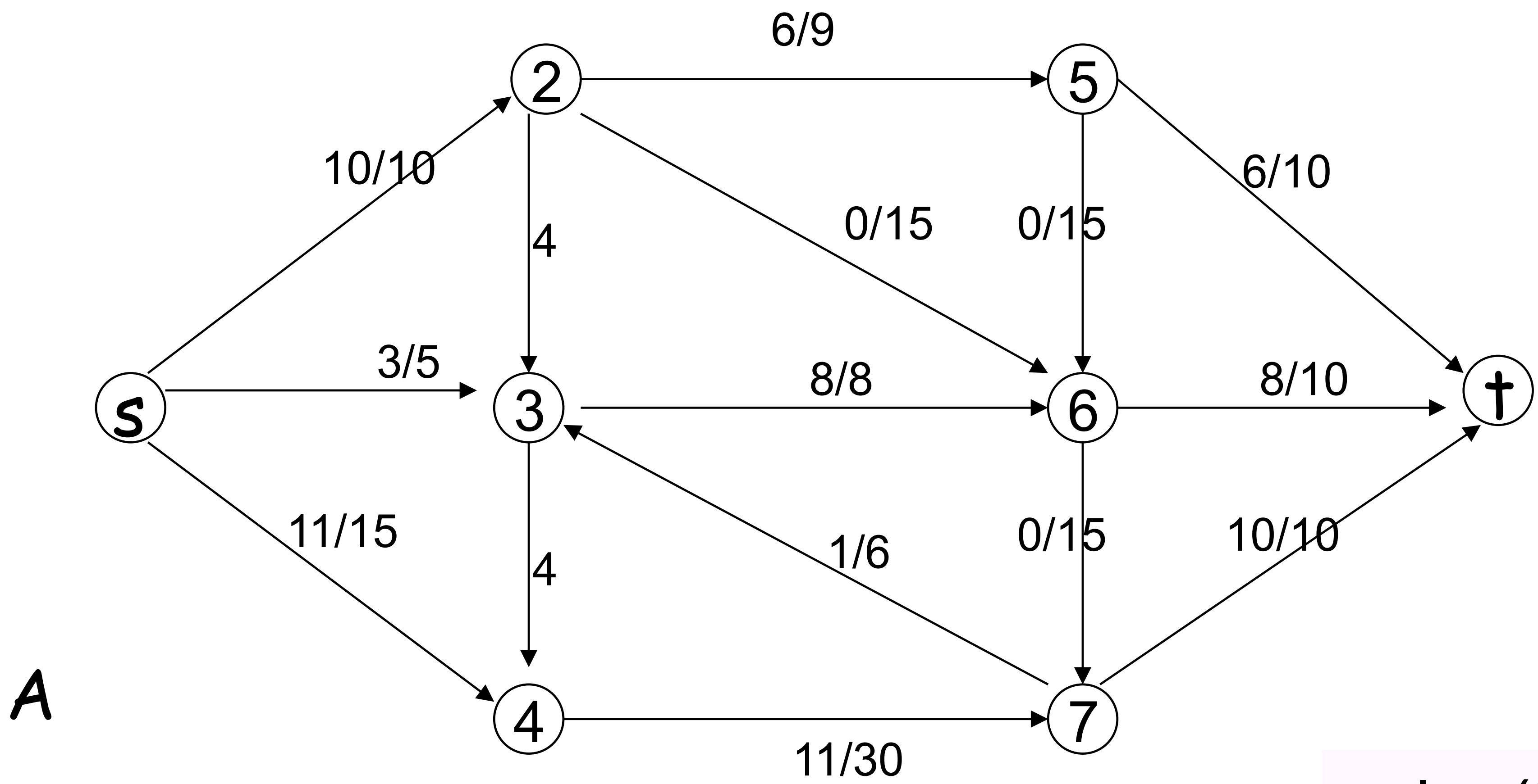


value(f) = 24

Example: Pick any cut yourself

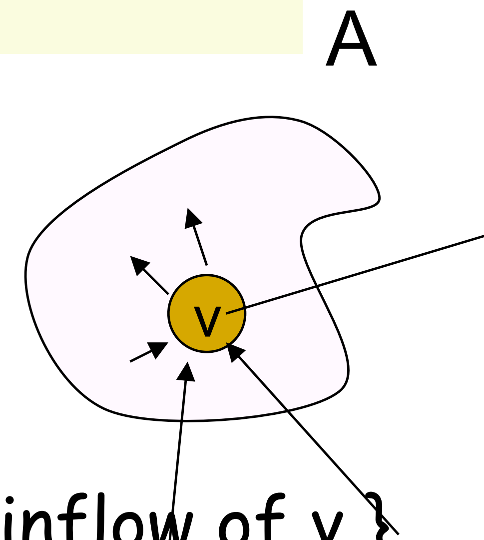
Let f be any flow of G, and let (A,B) be any s-t cut of G. Then value(f) = $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$.

Net flow =



value(f) = 24

Proof of flow value lemma



To prove: value(f) = $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$.

Proof.

Claim 1: value(f) = $\sum_{v \text{ in } A}$ { total outflow of v – total inflow of v }

- For any vertex v in A, except s, total inflow of v = total outflow of v.
- $\sum_{v \text{ in } A}$ { total outflow of v – total inflow of v }
  = total outflow of s
  = value(f)

Claim 2: $\sum_{v \text{ in } A}$ { total outflow of v } = $\sum_{e \text{ inside } A} f(e) + \sum_{e \text{ out of } A} f(e)$.

Claim 3: $\sum_{v \text{ in } A}$ { total inflow of v } = $\sum_{e \text{ inside } A} f(e) + \sum_{e \text{ into } A} f(e)$.

Claim 2 II

To prove: $\Sigma_{v \text{ in } A}$ { total outflow of v } = $\Sigma_{e \text{ inside } A}$ f(e) + $\Sigma_{e \text{ out of } A}$ f(e).

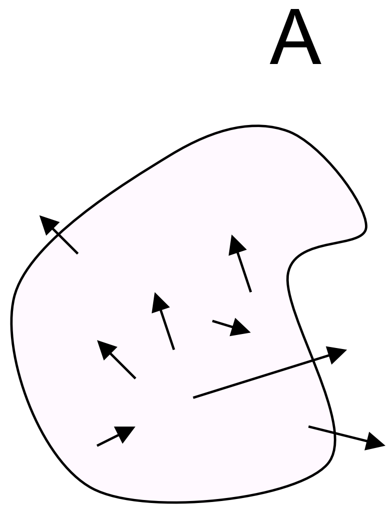
Proof.

- $\Sigma_{v \text{ in } A}$ { total outflow of v }

  = $\Sigma_{v \text{ in } A} \Sigma_{e \text{ is from } v}$ f(e)

  = $\Sigma_{e \text{ is from a vertex in } A}$ f(e)

  = $\Sigma_{e \text{ inside } A}$ f(e) + $\Sigma_{e \text{ out of } A}$ f(e)



Claim 3

II

To prove: $\Sigma_{v \text{ in } A}$ { total inflow of v } = $\Sigma_{e \text{ inside } A}$ f(e) + $\Sigma_{e \text{ into } A}$ f(e).

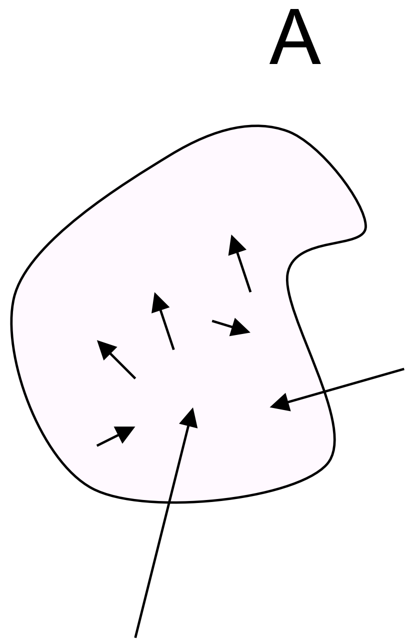
Proof.

$\Sigma_{v \text{ in } A}$ { total inflow of v }

= $\Sigma_{v \text{ in } A} \Sigma_{e \text{ into } v}$ f(e)

= $\Sigma_{e \text{ goes into a vertex in } A}$ f(e)

= $\Sigma_{e \text{ inside } A}$ f(e) + $\Sigma_{e \text{ into } A}$ f(e)



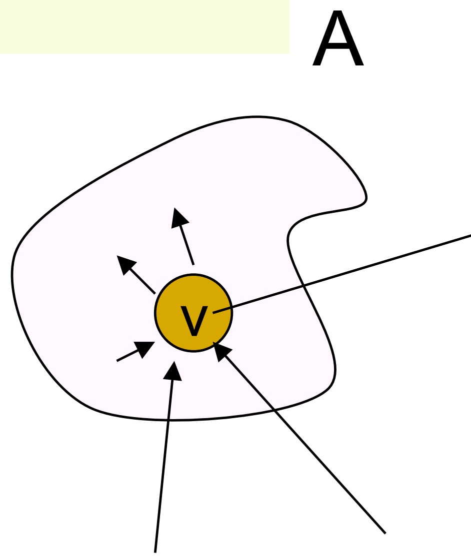
Summary: flow value lemma

- value(f) = $\sum_{v \text{ in } A}$ { total outflow of v – total inflow of v }
- $\boldsymbol{\Sigma}_{\text{in } A}$ { total outflow of v – total inflow of v }

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

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

Summary: flow value lemma



To prove: value(f) = $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$.

Proof.

Claim 1: value(f) = $\sum_{v \text{ in } A}$ { total outflow of v – total inflow of v }

Claim 2: $\sum_{v \text{ in } A}$ { total outflow of v } = $\sum_{e \text{ inside } A} f(e) + \sum_{e \text{ out of } A} f(e)$.

Claim 3: $\sum_{v \text{ in } A}$ { total inflow of v } = $\sum_{e \text{ inside } A} f(e) + \sum_{e \text{ into } A} f(e)$.

Corollary of flow value lemma.

Let (A,B) be any s-t cut. Let f be any flow.

Lemma (flow value Lemma).

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

Corollary 1. value(f) ≤ cap(A,B) $\quad (\text{cap}(A,B) = \sum_{e \text{ out of } A} c(e)).$

Proof.

$$\text{value}(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$
$$\leq \sum_{e \text{ out of } A} f(e) \leq \sum_{e \text{ out of } A} c(e) = \text{cap}(A,B).$$

Another corollary

Let (A,B) be any s-t cut. Let f be any flow.

Lemma (flow value Lemma).

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

Corollary 1. value(f) ≤ cap(A,B).

Corollary 2. If value(f*) = cap(A,B), then f* is a max flow.

<u>Proof</u>.

Let f' be any other flow. By Corollary 1, value(f') ≤ cap(A,B) = value(f*).

Organization

Definition of a cut.

Flow value lemma.

⇒ For all possible flow f′, value(f′) ≤ the capacity of a "cut".

⇒ If f^* admits no augmenting paths, then the network has a "cut" of with capacity = value(f^*).

Augmenting path theorem

Theorem 3. Let f be a flow such that G_f has no augmentation paths. Then there exists an s-t cut (A,B) such that value(f) = cap(A,B).

Proof of Theorem 3

Let f be a flow with no augmentation paths.

Consider the residual network G_f

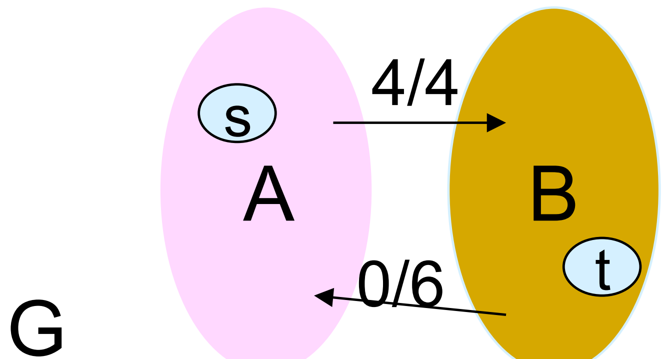
- It contains forward & backward edges with +ve edge capacity.

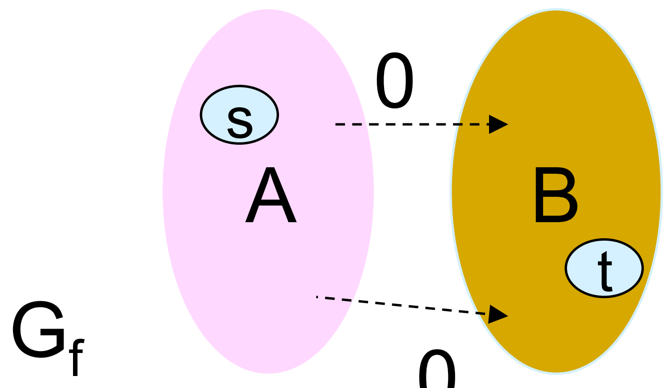
G_f has no path from s to t.

Let A be the set of vertices reachable from s in G_f.

- Let B = V – A.   And t must be in B.

G_f : no edge from A to B with +ve residual capacity.





Proof of Theorem 3

That means, w.r.t. G and f,

- every edge $e=(u,v)$ from A to B has a flow = full capacity and hence the forward edge (u,v) in G_f has zero residual capacity;
- every edge $e'=(w,x)$ from B to A has zero flow and hence the backward edge (x,w) from B to A has zero residual capacity.

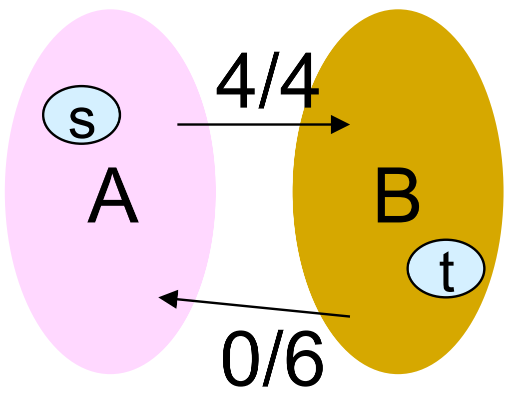
By flow value lemma,

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

$= \sum_{e \text{ out of } A} f(e) - 0$

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

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



Conclusion

Flow value lemma (slides 33-38)

$\Rightarrow$ For all possible flow f', value(f') ≤ the capacity of a "cut".

$\Rightarrow$ If *f** admits no augmenting paths, then the network has a "cut" of with capacity = value(*f**).

Therefore, *f** is a maximum flow.

Max flow = Min Cut

maximum flow $\leq$ minimum cut (capacity).

Because, by Corollary 1, value(f) $\leq$ cap(A,B) for any flow f and cut (A,B).

maximum flow $\geq$ minimum cut.

By Theorem 3, value(f) = cap(A,B), where f is a flow admitting no augmentation paths, and (A,B) is a cut.

Therefore, max flow = value(f) = cap(A,B) $\geq$ min cut.