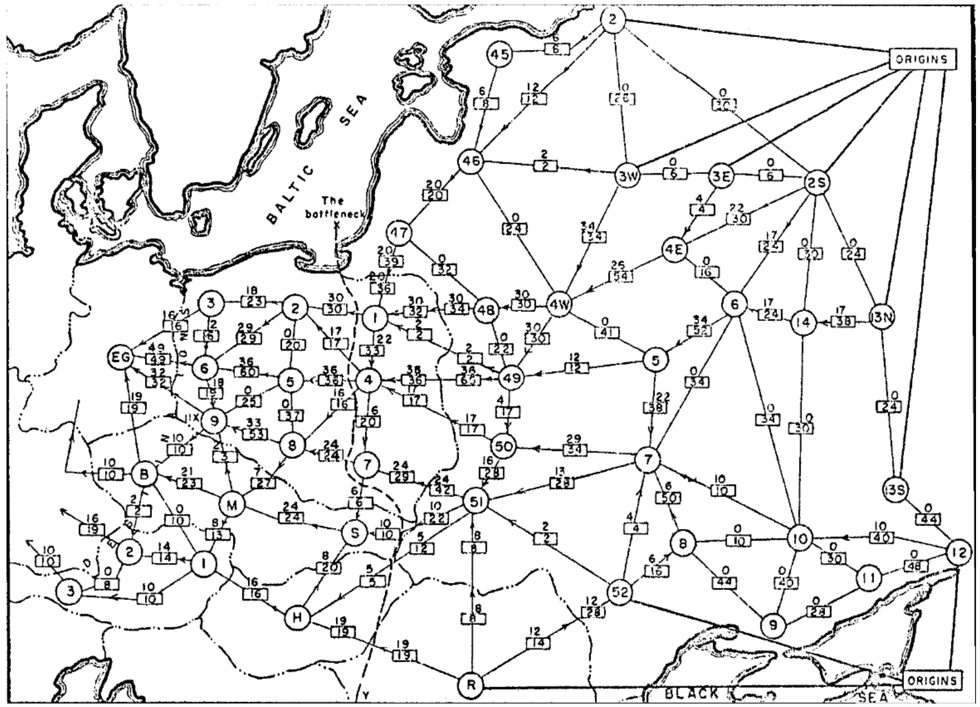
Lecture 30: Network Flow III



COSC 311 *Algorithms,* Fall 2022

Annoucement

Midterm II on Wednesday

- Practice solutions coming soon

Last Time

Max Flow Problem:

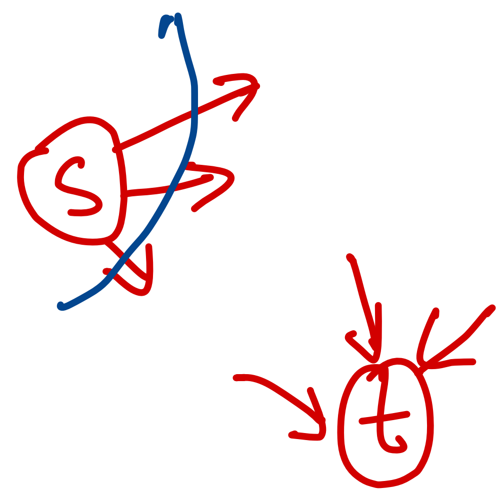
Input.

- weighted directed graph $G = (V, E)$
  - weights = edge capacities > 0
- source s, sink t
  - all edges oriented out of s
  - all edges oriented into t

Output.

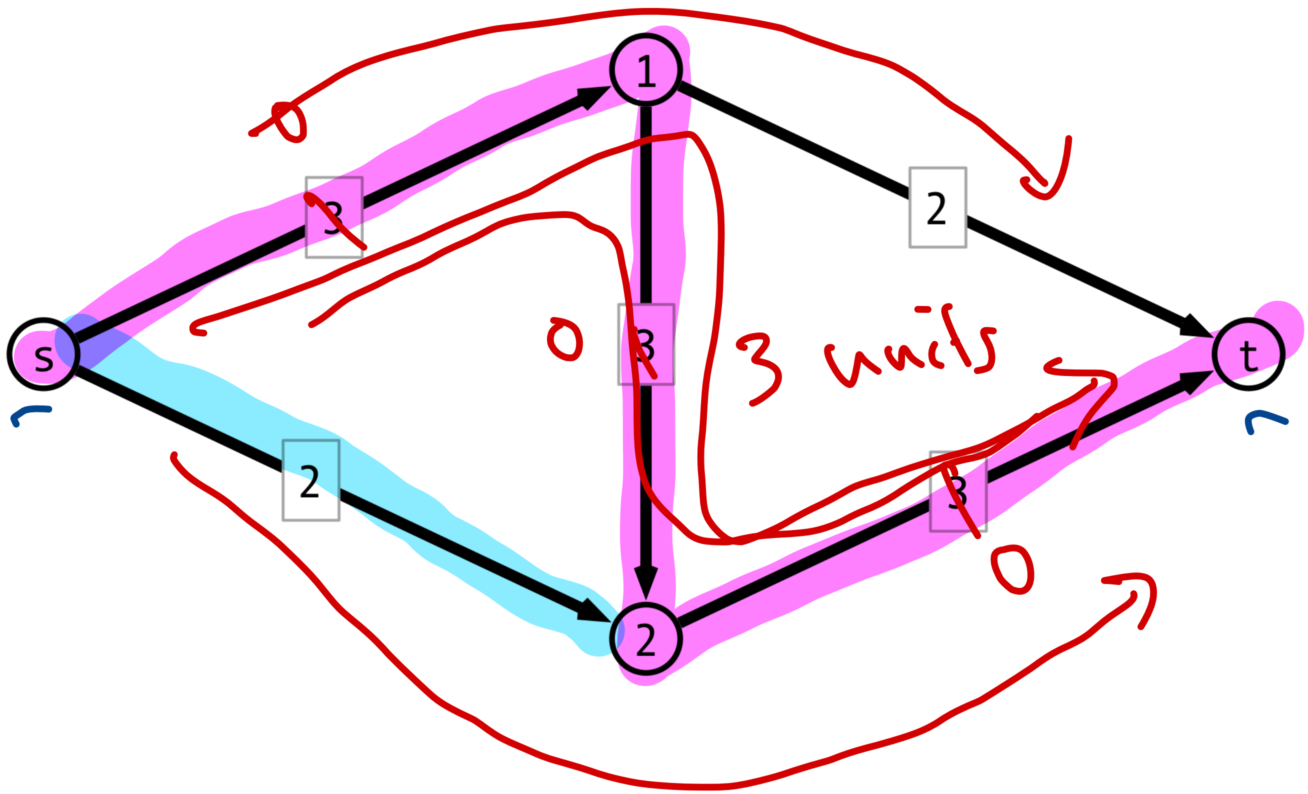
- flow f of maximum value
  - $\text{val}(f) = \sum_{s \to v} f(s, v)$

Val(f) = sum of flow from s

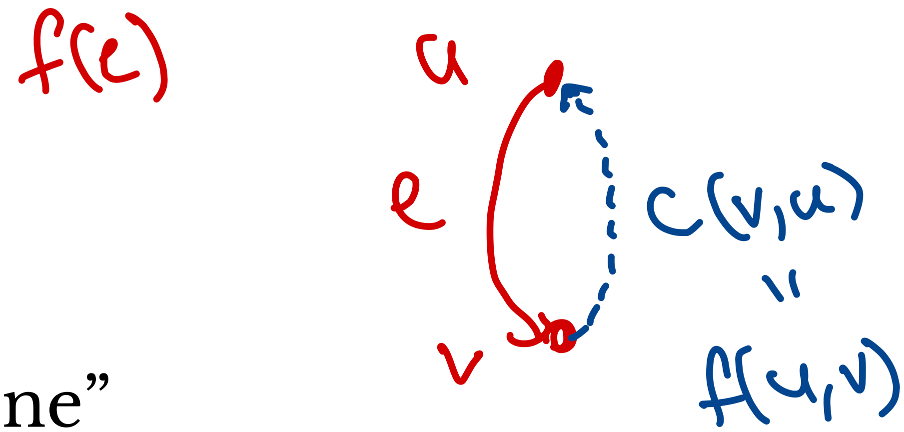


We Showed

Greedy strategy doesn't always work



Ford-Fulkerson Idea



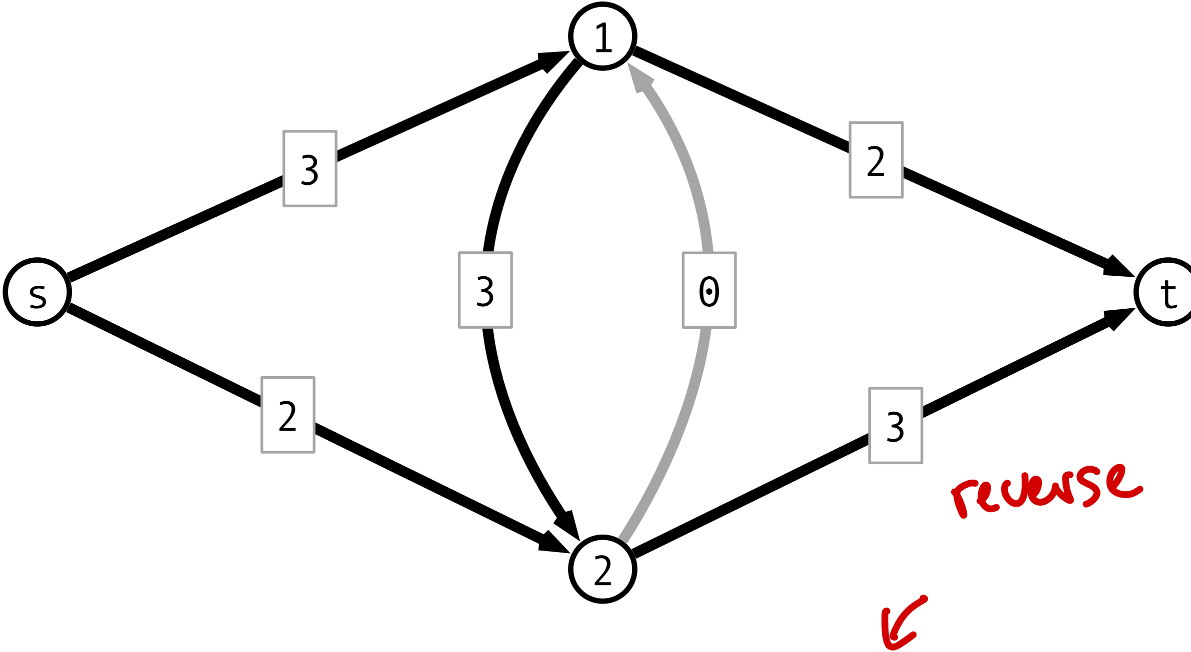
Given a flow f:

- allow forward flow to be "undone"
- when routing forward flow $f(u, v)$ across edge (u, v), create backwards edge (v, u) with capacity $f(u, v)$
    - graph with backwards edges = *residual graph*
- backwards flow cancels out forward flow

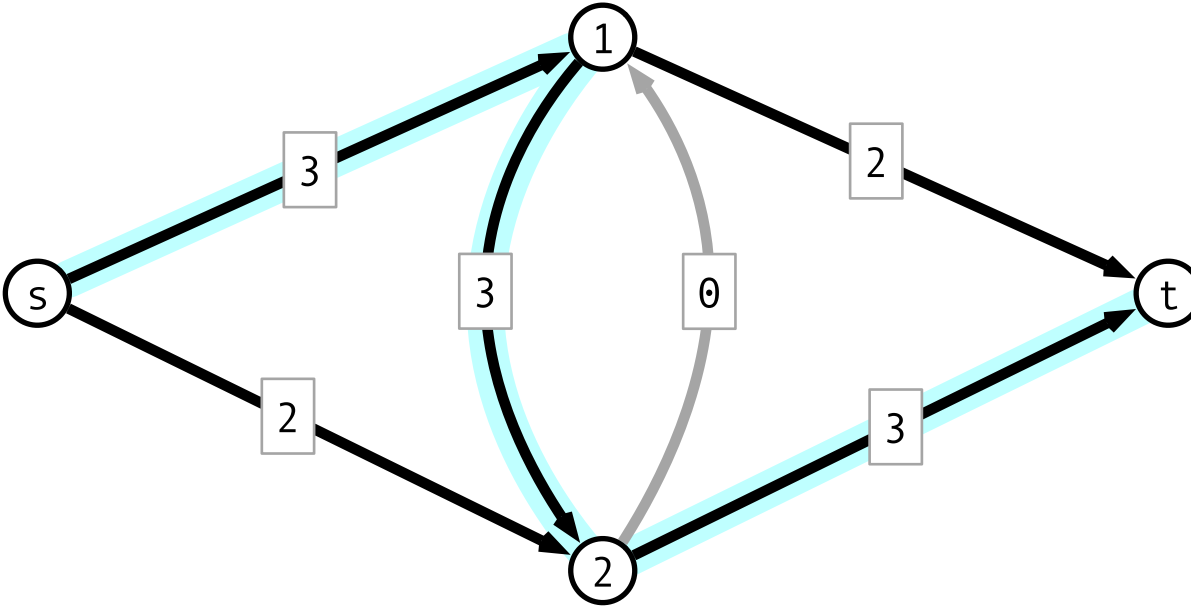
Ford-Fulkerson Algorithm:

1. apply greedy strategy to residual graph
2. update residual graph with new flow
3. continue until no unsaturated path from s to t remains

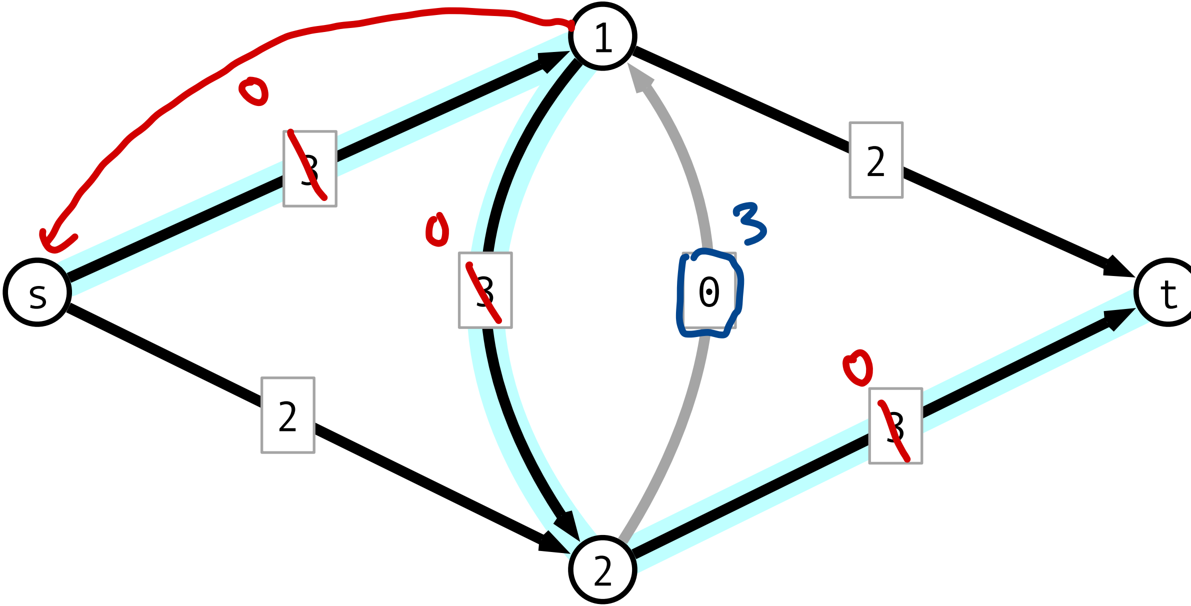
Ford-Fulkerson Example



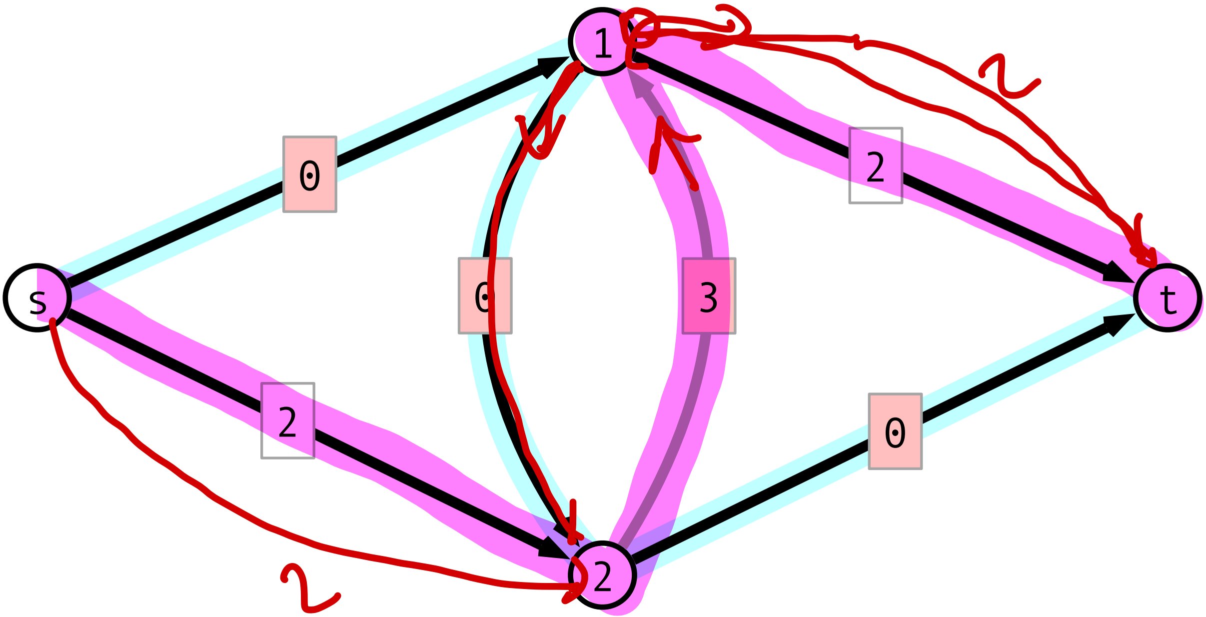
e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	0	0	0	0	0	0



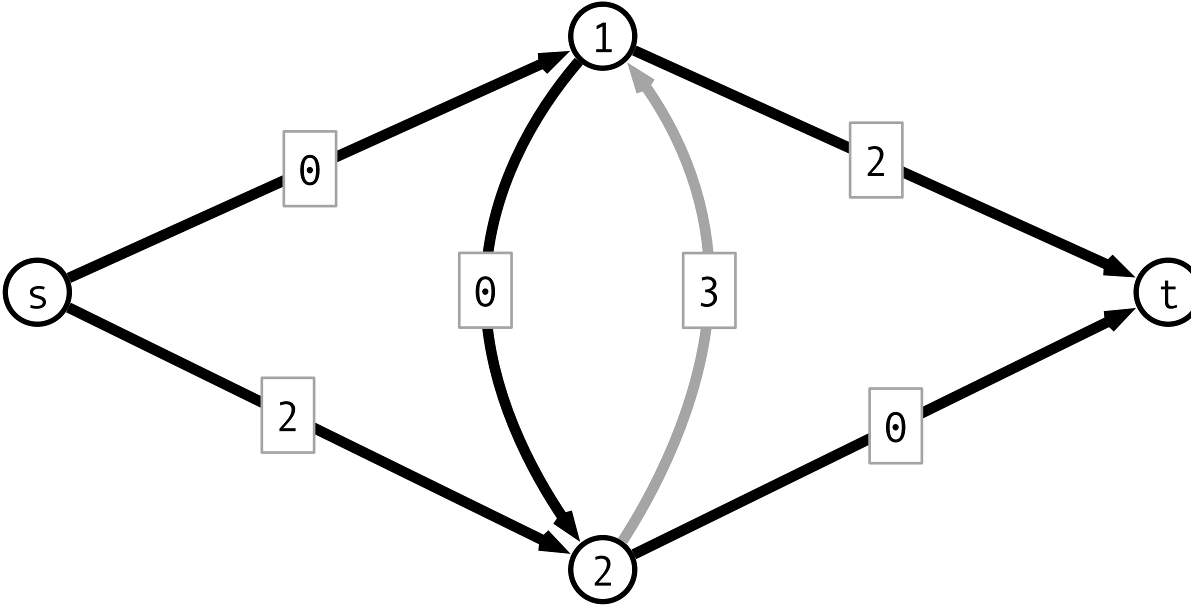
e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	0	0	0	0	0	0



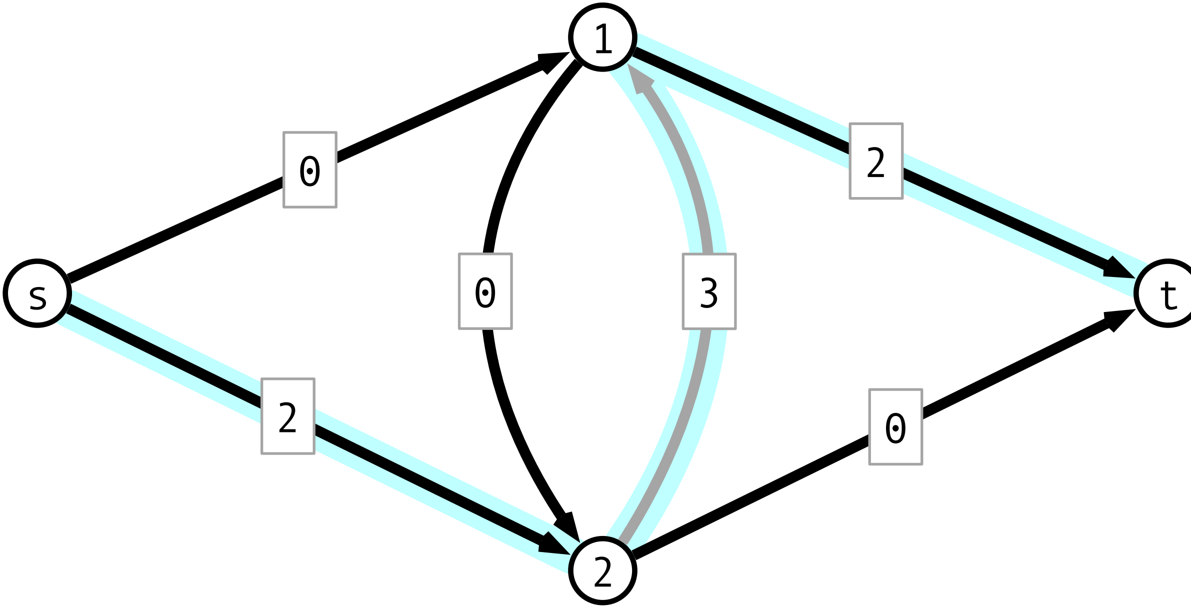
e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	0	3	0	0	3



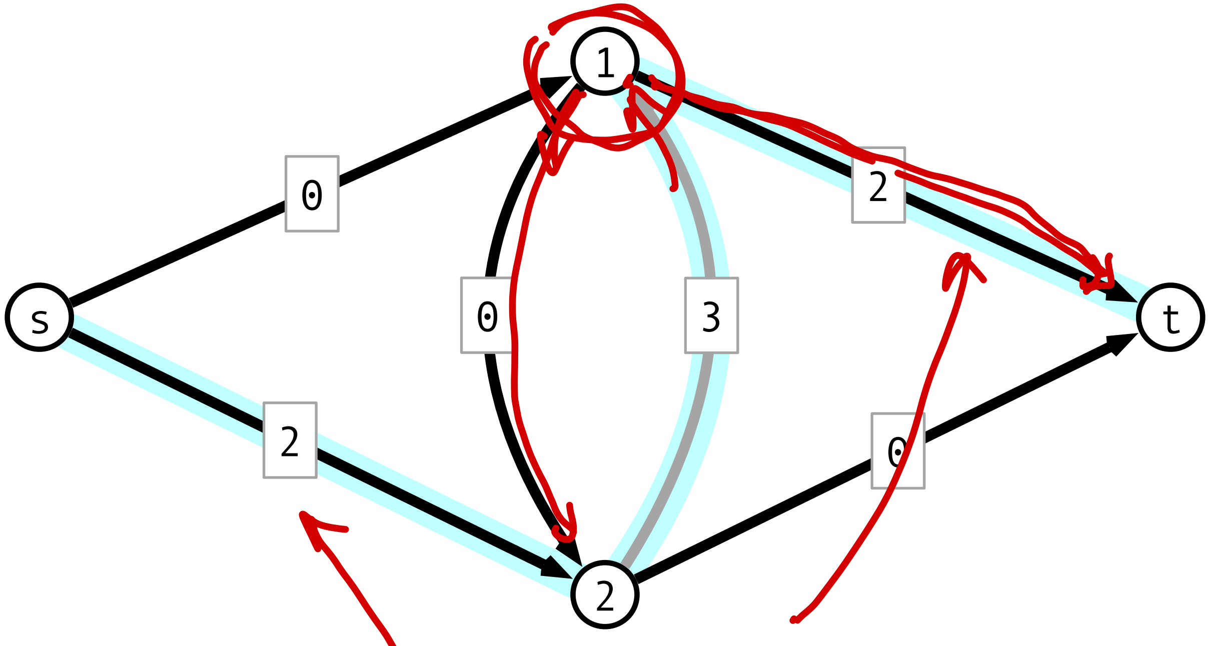
e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	0	3	0	0	3



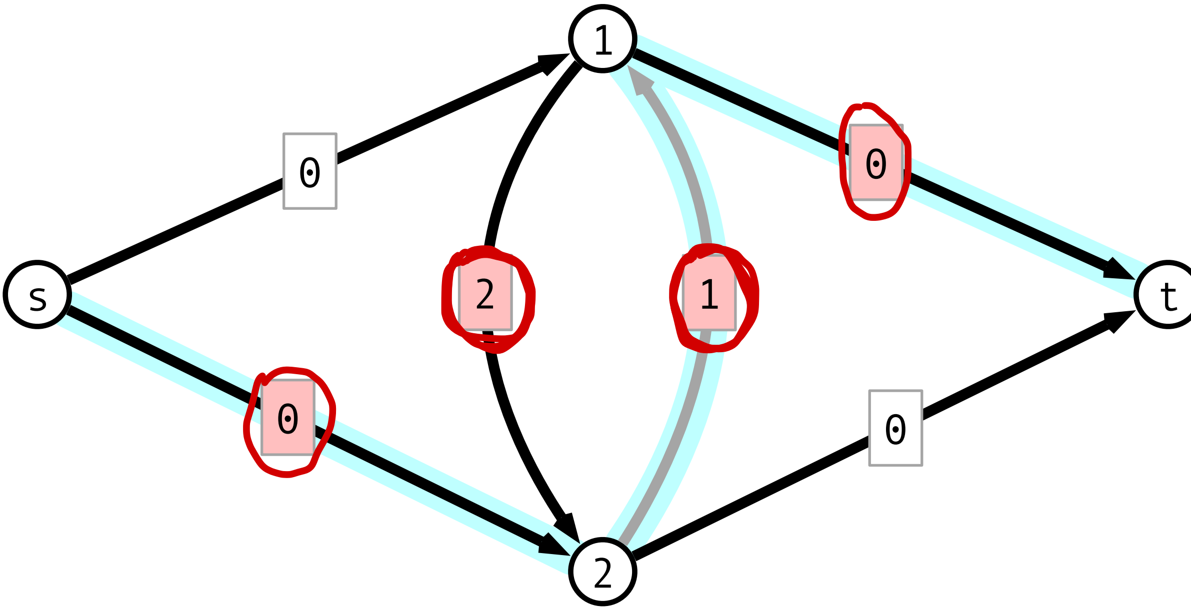
e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	0	3	0	0	3



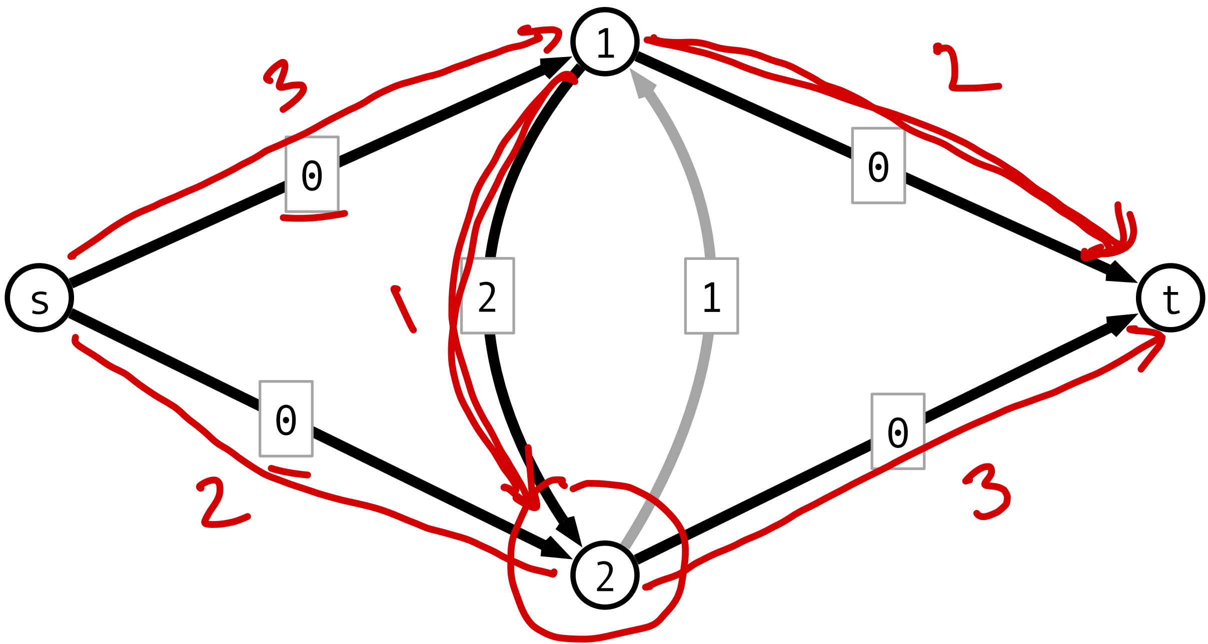
e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	0	3	0	0	3



e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	2	1	2	0	3



e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	2	1	2	0	3



e	(s, 1)	(s, 2)	(1, 2)	(1, t)	(2, 1)	(2, t)
f(e)	3	2	1	2	0	3

Questions

How do we...

1. find *augmenting path* P from s to t?

   DFS, BFS, ...

   b = min remaining capacity along path

2. update flow f according to P?

   (u,v) is forward edge, increment f(u,v) by b

   (u,v) is backward edge: decrement f(v,u) by b

   (v,u): corresp. fwd edge

3. update residual graph G_f?

   (u,v) is fwd edge:
   - decrement cap(u,v) by b
   - increment cap(v,u) by b

   (u,v) is backward edge: same!

Formalizing Ford-Fulkerson

```
MaxFlow(G, s, t):
  Gf <- G
  f <- zero flow
  P <- FindPath(Gf, s, t)
  while P is not null do:
    b <- min capacity of any edge in P
    Augment(Gf, f, P, b)
    P <- FindPath(Gf, s, t)
  endwhile
  return f
```

residual graph

flow

BFS

update flow, resid. graph

Augment Procedure

```
Augment(Gf, f, P, b):
  for each edge (u, v) in P
    if (u, v) is forward edge then
      f(u, v) <- f(u, v) + b
      c(u, v) <- c(u, v) - b
      c(v, u) <- c(v, u) + b
    else
      f(v, u) <- f(v, u) - b
      c(v, u) <- c(v, u) + b
      c(u, v) <- c(u, v) - b
```

Running Time

n = # nodes
m = # edges

Assume:

1. all capacities are integers
2. C = sum of capacites of edges out of s

Observe:

1. How long to find augmenting path P?

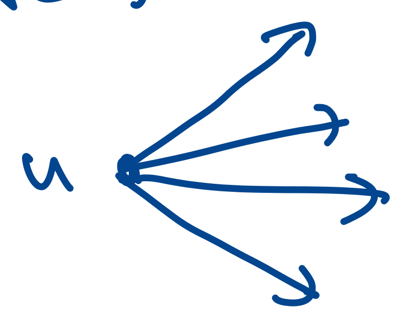
   $O(m)$

2. How long to run `Augment`?

   $O(m)$ (or $O(n)$)

3. How many iterations of find/augment?

   $\leq C$ bc each iter. increases val by ≥ 1

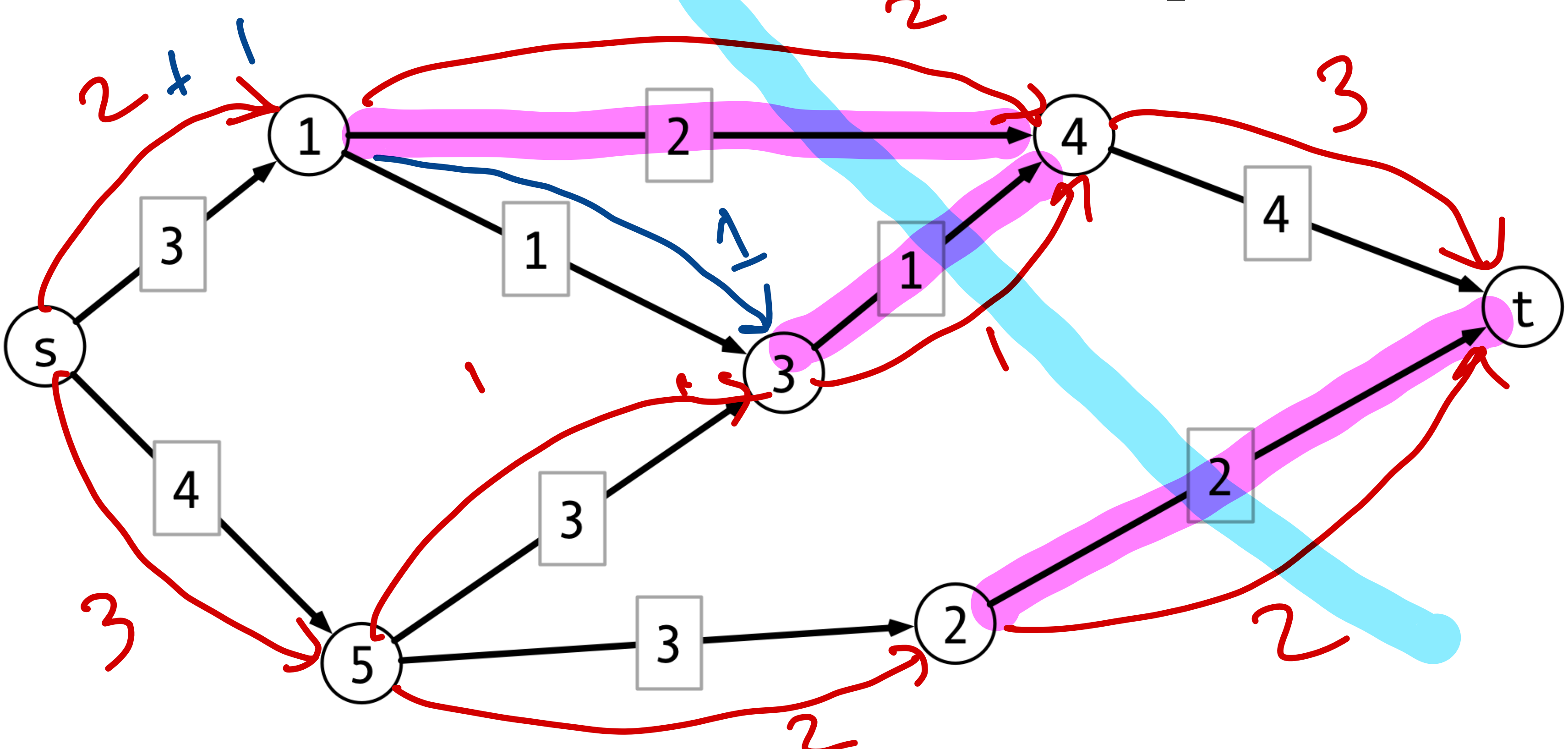


Conclude: Overall running time?

$O(C \cdot m)$

Optimality of Flow?

Question. How do we know this flow is optimal?



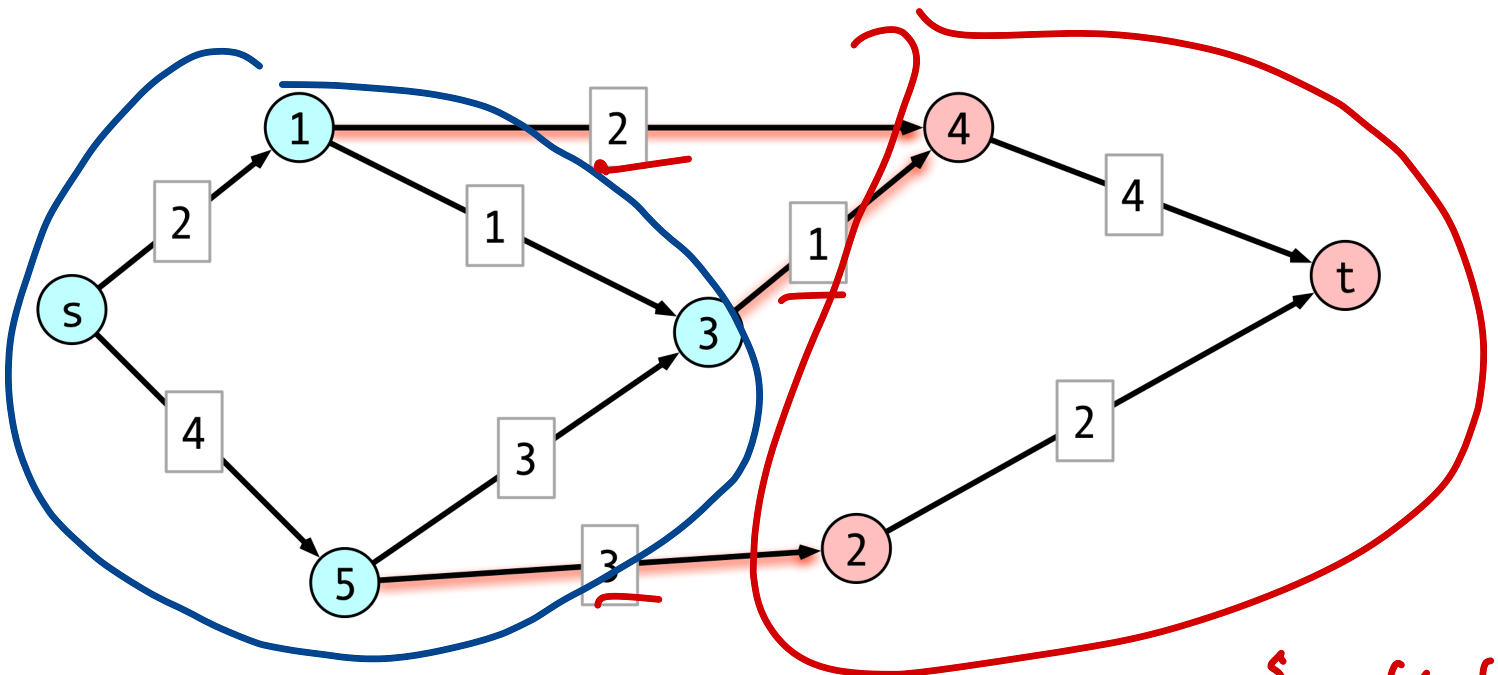
e	s,1	s,5	1,3	1,4	2,t	3,4	4,t	5,3	5,2
f(e)	2	3		2	2	1	2	1	2

Cuts

Definition. An s - t **cut** (A, B) is a partition of vertices into two disjoint sets with s in A and t in B.

The **capacity** of (A, B), denoted $\text{cap}(A, B)$ is the sum of the capacities of the edges out of A.

Cut Example



A = {s, 1, 3, 5}

B = {2, 4, t}

cap(A,B) = 2+1+3
= 6.

e	s , 1	s , 5	1 , 3	1 , 4	2 , t	3 , 4	4 , t	5 , 3	5 , 2
f(e)	2	3	0	2	2	1	3	1	2

Correctness of Ford-Fulkerson

Idea. Relate values of flows to capacities of cuts:

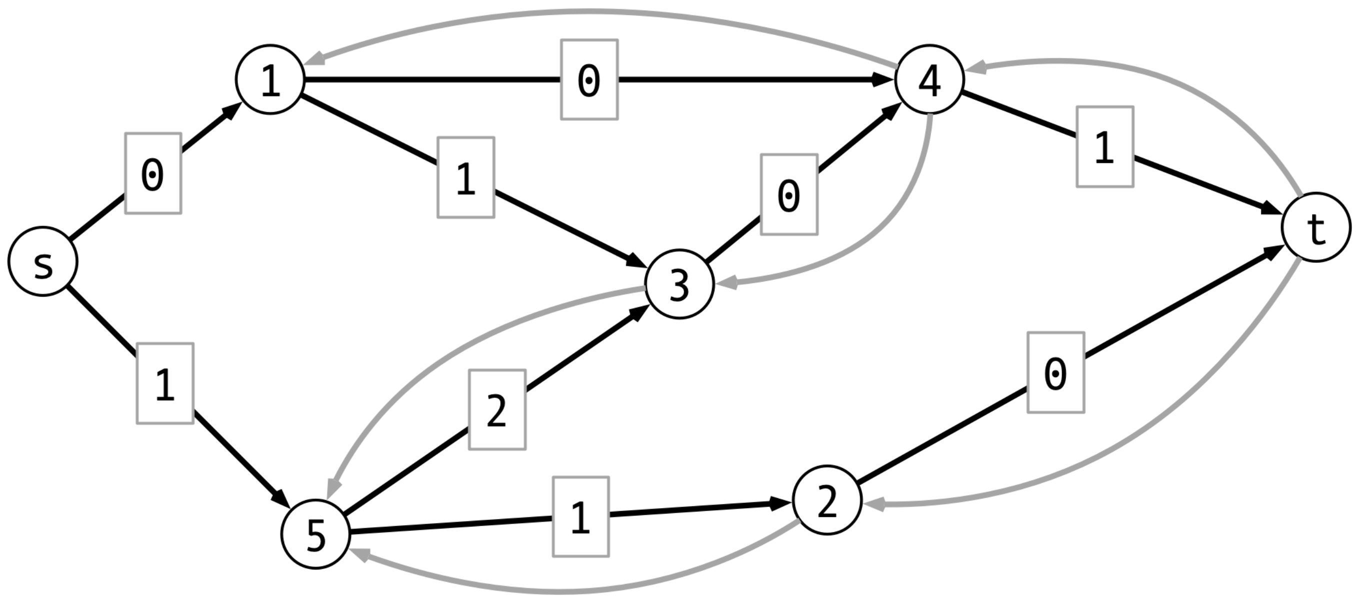
- max flow = min cut

Outline:

- for any cut (A, B), net flow across cut = value of flow
  - $\implies$ max flow $\leq$ min cut
- if f has no augmenting path in residual graph, then there is a cut with net flow = value of cut
  - $\implies$ value of f = capacity of cut
  - $\implies$ $\text{val}(f) = \text{cap}(A, B) \geq$ min cut

Together these imply Ford-Fulkerson produces max flow

Max Flow/Min Cut Example



e	s,1	s,5	1,3	1,4	2,t	3,4	4,t	5,3	5,2
f(e)	2	3	1	1	2	1	2	1	2

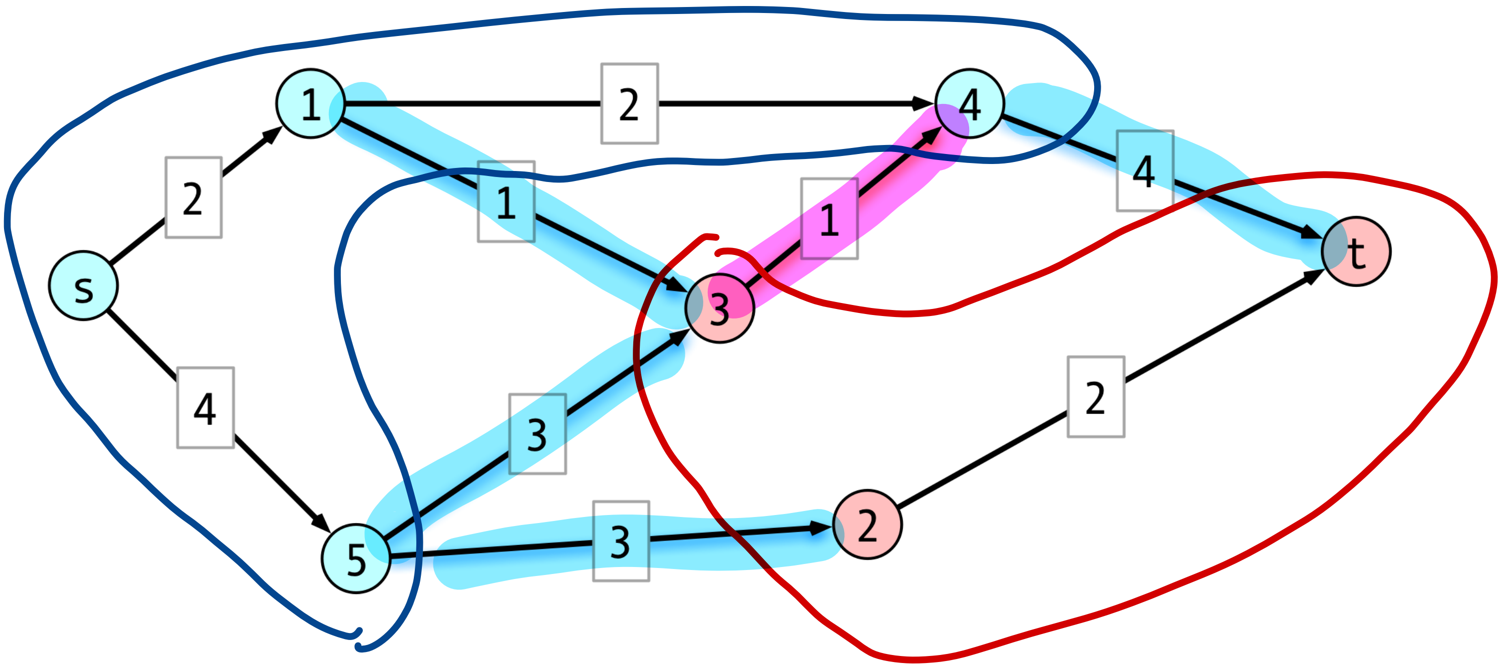
Claim 1

For any s - t cut (A, B) and flow f, $\text{val}(f) = f^{\text{out}}(A) - f^{\text{in}}(A)$

- $f^{\text{out}}(A) =$ flow out of A
- $f^{\text{in}}(A) =$ flow into A

Consequence. For all cuts (A, B), $\text{val}(f) \leq \text{cap}(A, B)$

Claim 1 Illustration



e	s,1	s,5	1,3	1,4	2,t	3,4	4,t	5,3	5,2
f(e)	2	3	0	2	2	1	3	1	2

2+3 = 5

6 out

1 in

5

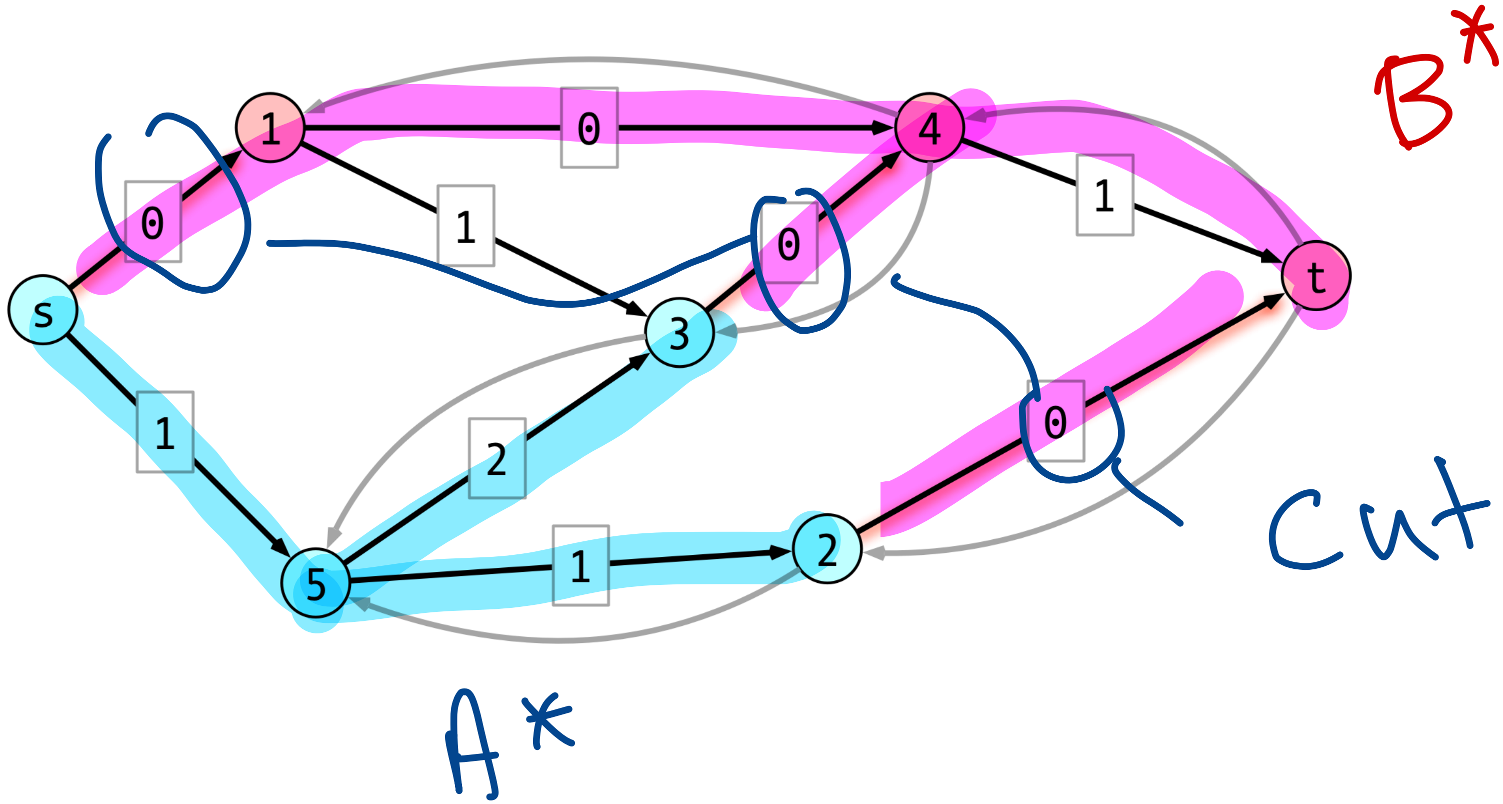
Claim 2

Suppose f does not have an augmenting path in the auxiliary graph.

- A^* = nodes reachable from s in auxiliary graph
- B^* = nodes not reachable

Then $\text{val}(f) = \text{cap}(A^*, B^\star)$

Claim 2 Illustration



e	s,1	s,5	1,3	1,4	2,t	3,4	4,t	5,3	5,2
f(e)	2	3	1	1	2	1	2	1	2

Correctness Follows

Consider flow f found by Ford-Fulkerson.

1. By claim 1, no flow can have value larger than any cut capacity
2. By claim 2, $\text{val}(f) = \text{cap}(A, B)$

These imply:

1. f is a maximum flow
2. (A, B) is a minimum cut

Conclusion

$G = (V, E)$ a weighted, directed graph with minimum cut capacity C.

Ford-Fulkerson finds maximum flow in time $O(Cm)$.

- can be modified to find minimum cut as well

Next Time

1. Midterm on Wednesday
2. Stable Matching + Will's research on Friday
3. Reductions and NP completeness after break