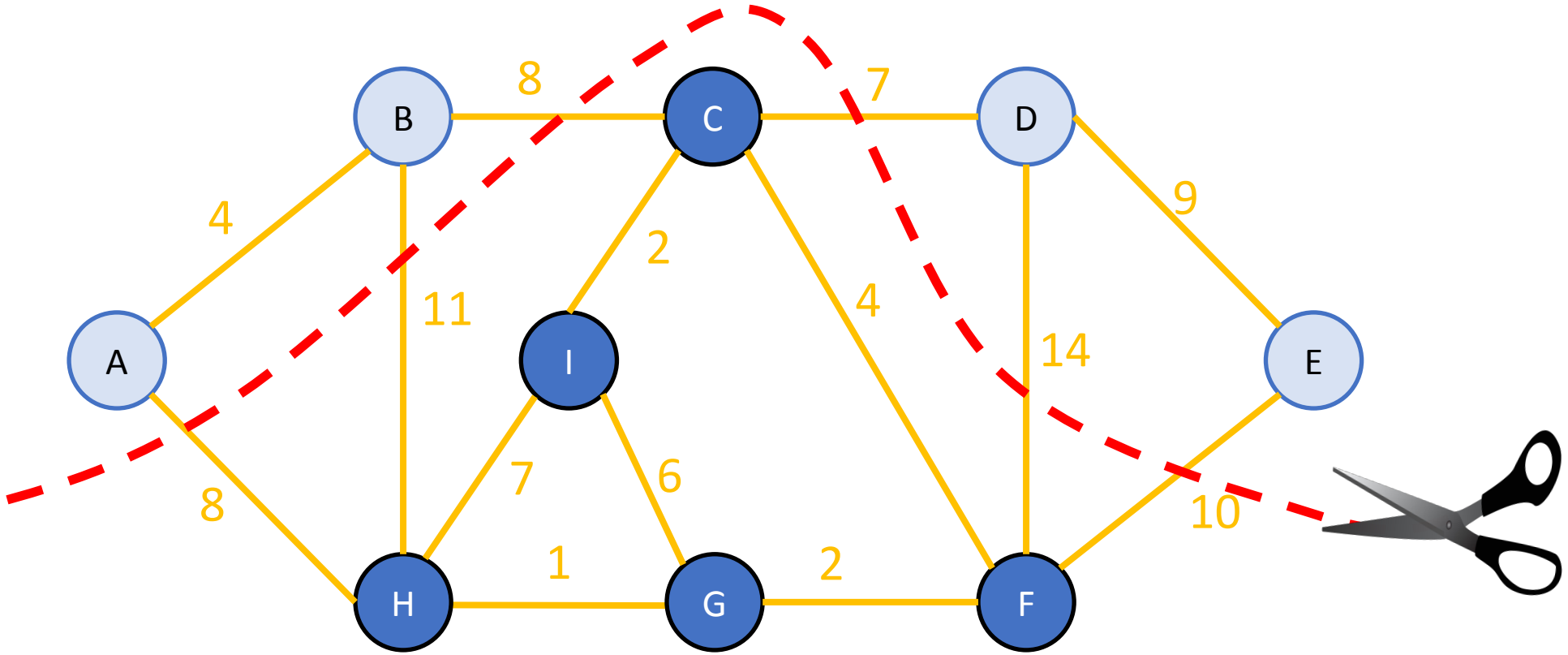
CS 161 Section 8

CA: [Name of CA]

s-t min-cut max-flow

Cuts in graphs

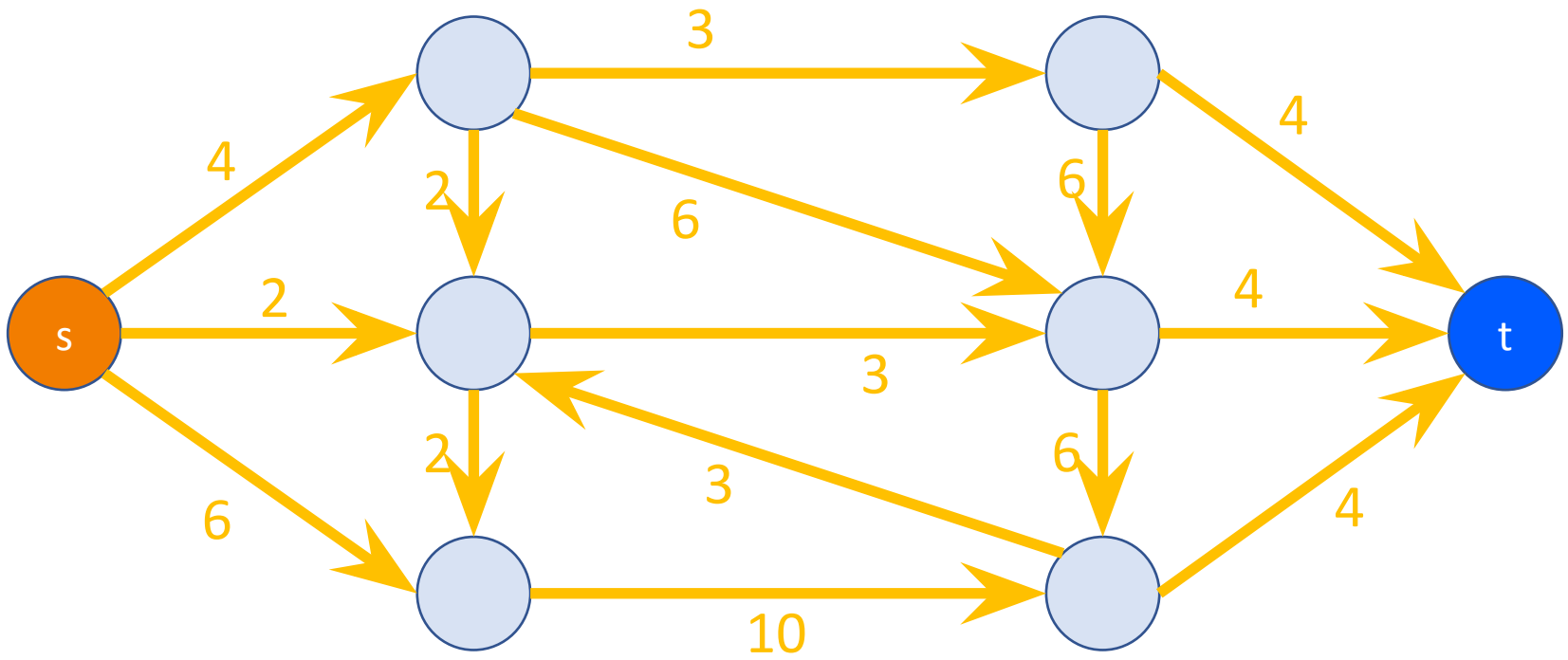
- A cut is a partition of the vertices into two parts:



This is the cut **"{A,B,D,E} and {C,I,H,G,F}"**

Today: s-t Cuts

- Graphs are directed and edges have "capacities" (weights)
- We have a special "source" vertex s and "sink" vertex t.
  - s has only outgoing edges*
  - t has only incoming edges*

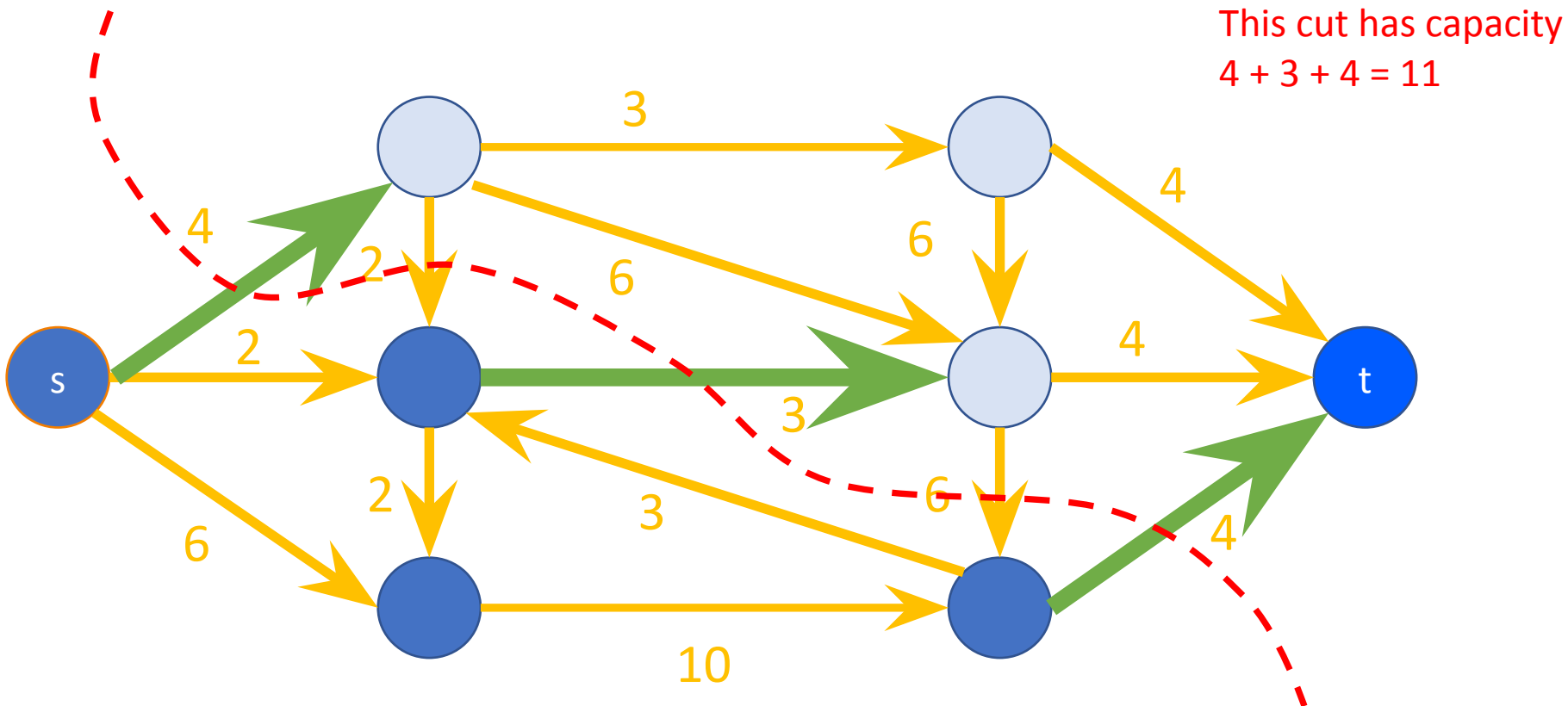


*at least for this class

A minimum **s-t cut**

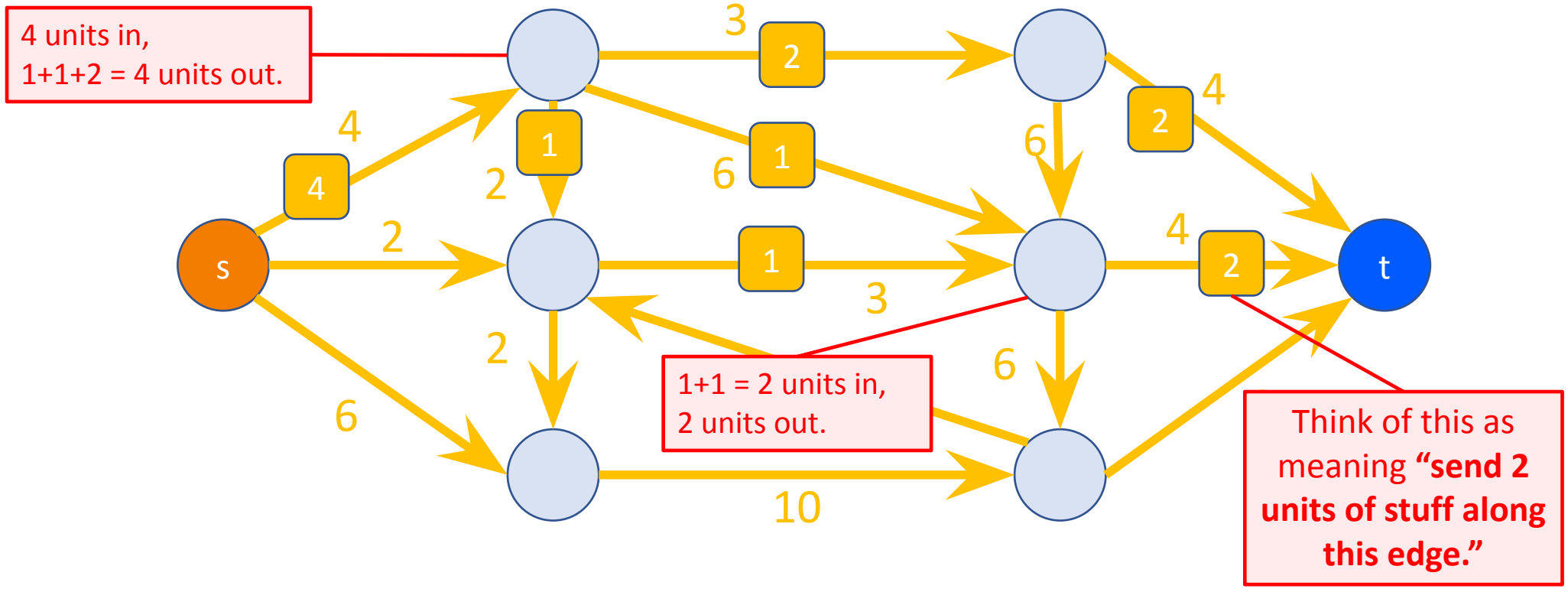
is a cut which separates s from t
with minimum capacity.

- Question: how do we find a minimum s-t cut?



Flows

- In addition to a capacity, each edge has a **flow**
  - (unmarked edges in the picture have flow 0)
- The flow on an edge must be less than or equal to its capacity.
- At each vertex other than s and t, the incoming flows must equal the outgoing flows.



But what is a “flow,” really?

- Think of s as a “water source” and t as a “water sink”
- Each edge is a pipe, and the capacity is the amount of water that can flow through the edge
- Clearly, for each pipe, as much water comes in must go out!
- Some pipes will be the “bottleneck” pipes… could these be related to the min-cut somehow?

Flows

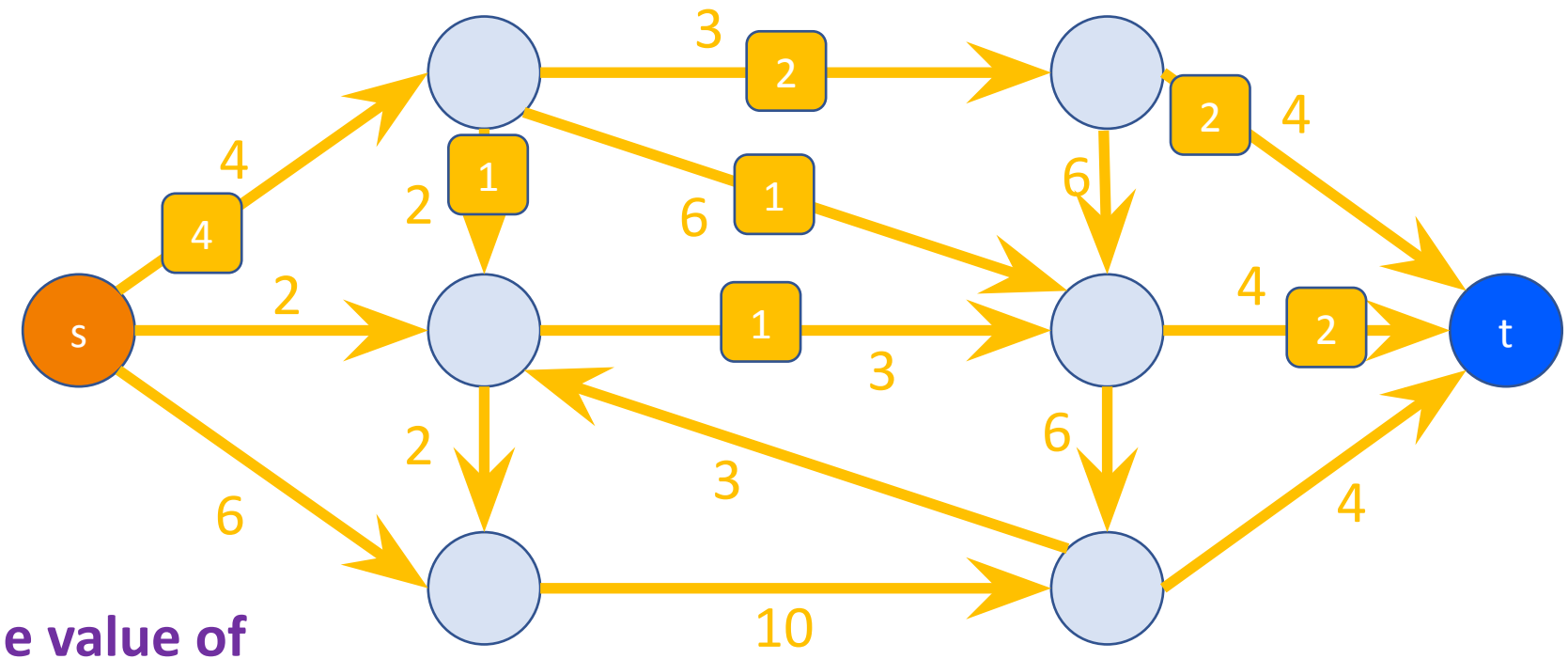
- The value of a flow is:
  - The amount of stuff coming out of s
  - The amount of stuff flowing into t
  - These are the same!

Because of conservation of flows at vertices,

stuff you put in

=

stuff you take out.



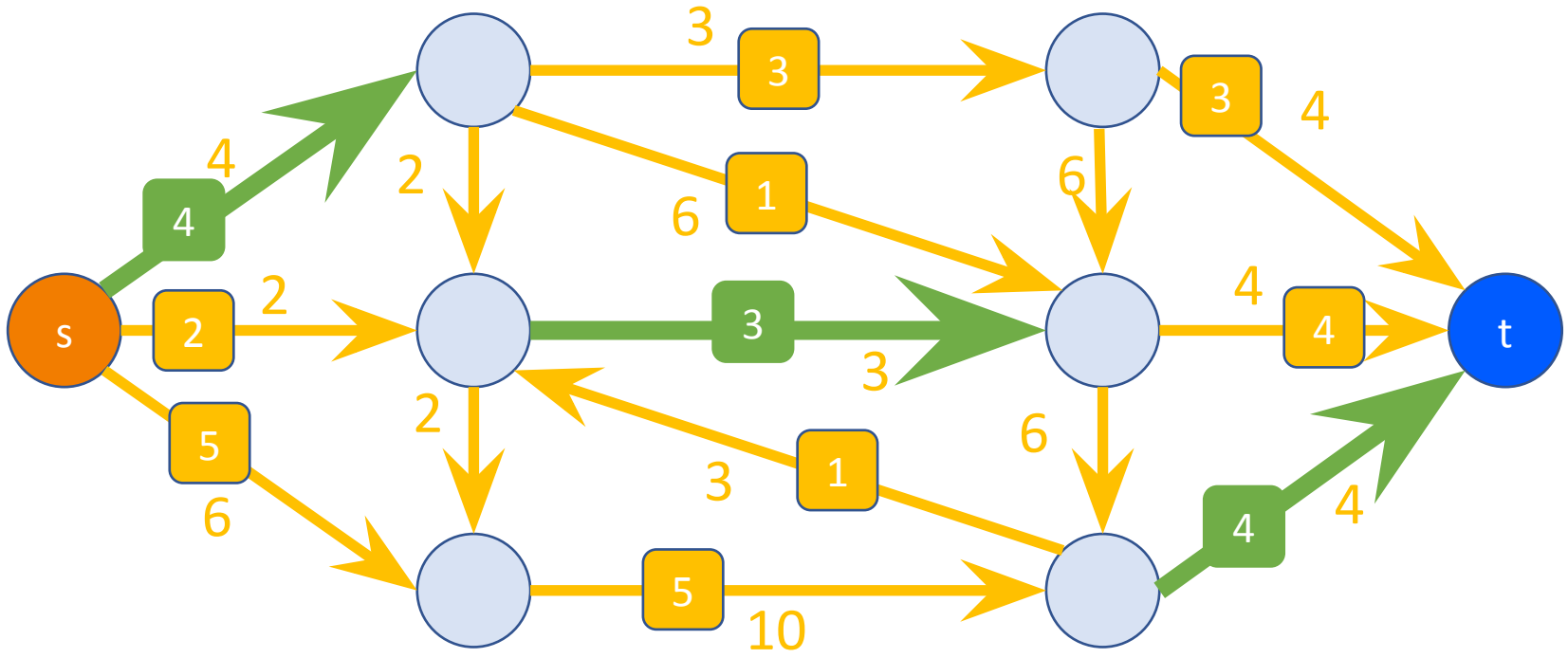
The value of this flow is 4.

Theorem

Max-flow min-cut theorem

The value of a max flow from s to t
is equal to
the capacity of a min s-t cut.

Intuition: in a max flow, the min cut better fill up, and this is the bottleneck.



Max-flow min-cut in water pipes

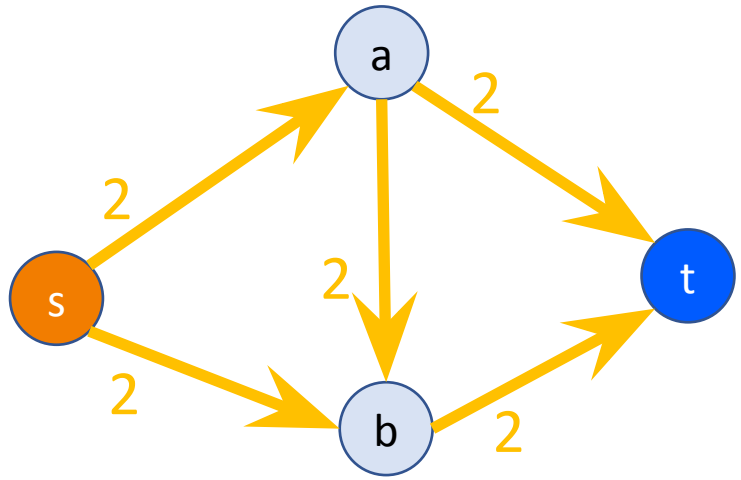
- What does the max-flow min-cut theorem mean in the context of our water pipes example before?
- Max-flow = maximum amount of water we can send
- Min-cut = the minimum capacity of pipes, which when removed, completely disconnects the water source and sink (i.e., no water flows)
- Intuitively makes sense!

Ford-Fulkerson algorithm

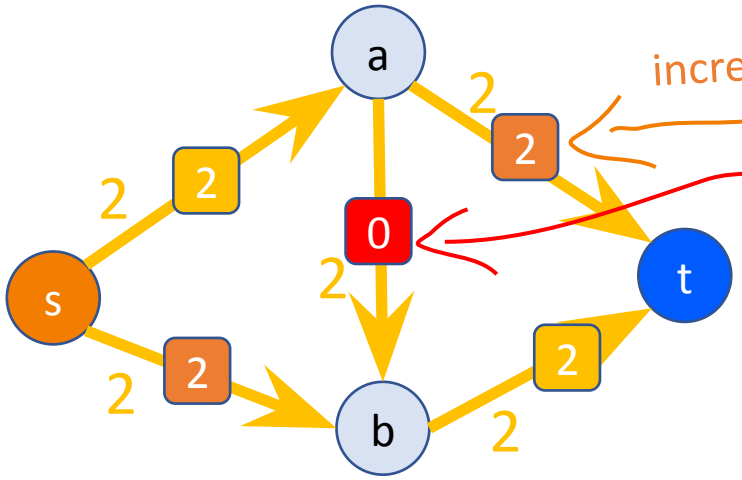
Outline of algorithm:

- Start with zero flow
- While we haven't found the max flow:
  - Find a way to send more from s to t

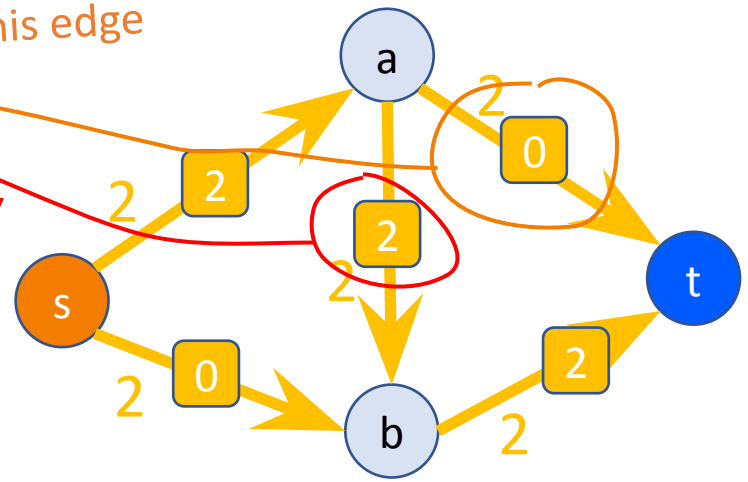
Original graph:



Iteration 3: optimal flow!



Iteration 2: increase the flow again:



increased flow on this edge

decreased flow on this edge

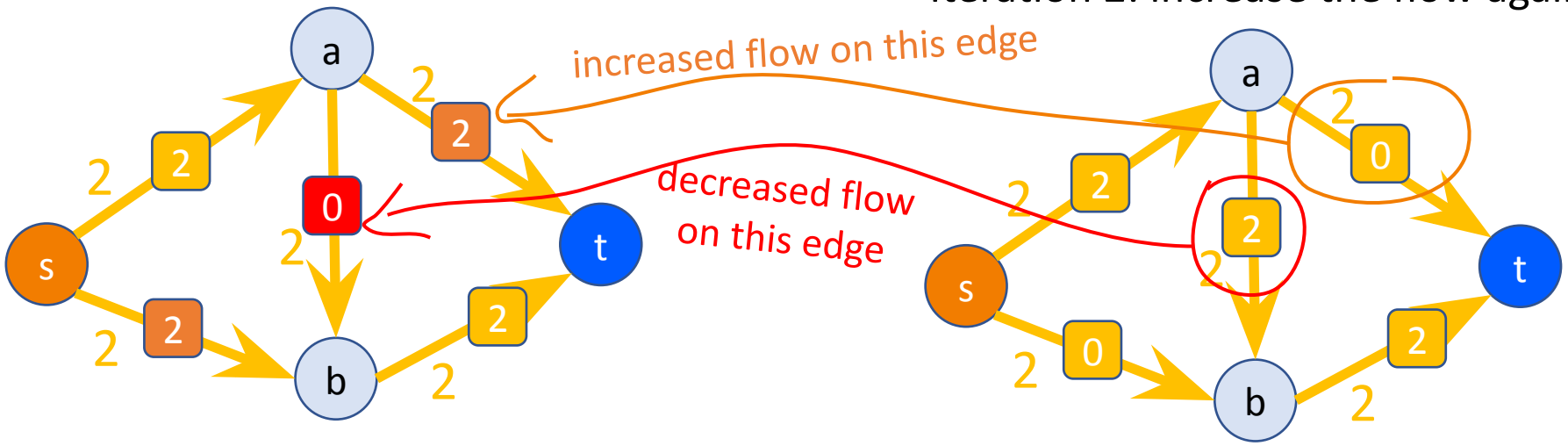
2 key ideas

1. In order to increase total flow, we *decreased* flow on some edges

2. Decreasing flow in one direction = increasing flow in the opposite direction
   = "sending back" this flow

Iteration 3: optimal flow!

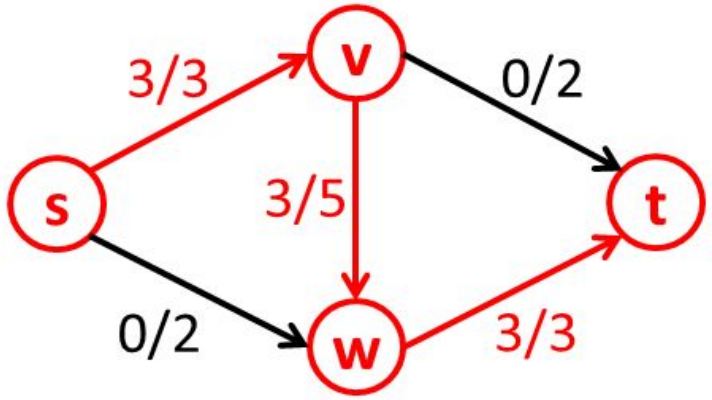
Iteration 2: increase the flow again:



Why residual graphs?

Let's think of a greedy idea for finding flows:

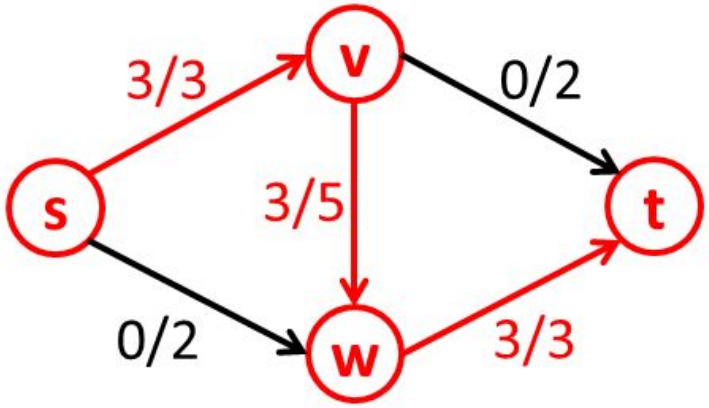
- Pick an arbitrary augmenting path, and push the maximum flow you can through this path
- Keep repeating this until no paths exist!
- We reach a "blocking" flow: no more paths exist, even though the flow is not optimal… what to do?

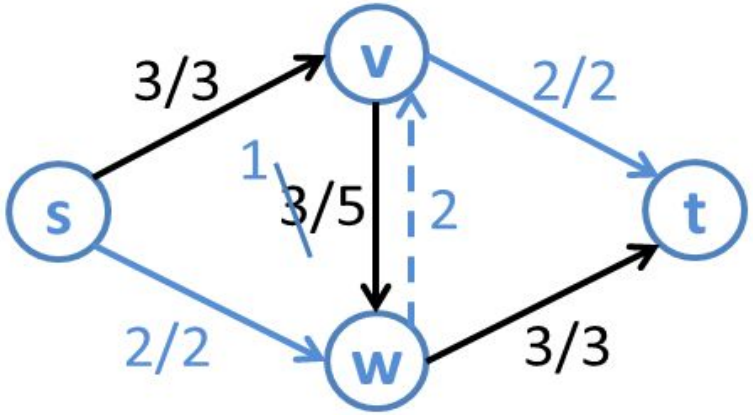


Reference: https://cs.stackexchange.com/a/71505

Why residual graphs?

- Fix this issue by allowing “undo” operations!
- Akin to reversing our previous decision in a greedy algorithm
- Encoding these operations is the main goal of a residual graph

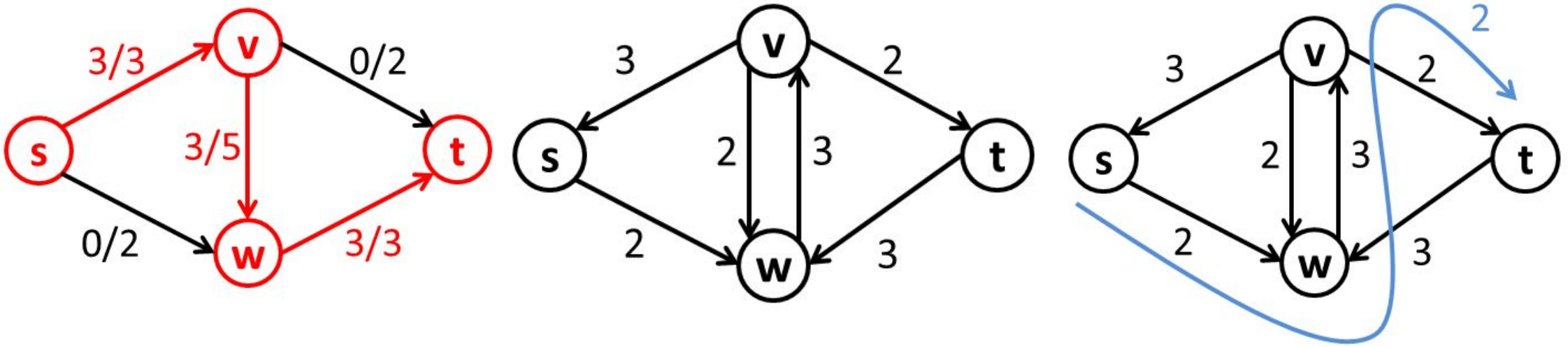




Reference: https://cs.stackexchange.com/a/71505

What is a residual graph?

- Stores how much flow can be pushed through an edge
- Having the reverse edge capacities = current flow represents that the current flow can be “undone!”



Reference: https://cs.stackexchange.com/a/71505

Time complexity of max flow?

- Suppose all capacities are integers, and let $|f|$ be the maximum flow in the graph.

- **Theorem**: If you find augmenting path using BFS or DFS, the Ford-Fulkerson algorithm runs in time $\boldsymbol{O(m|f|)}$.

- Proof:

  In each iteration, the flow increases by at least +1.

  $\Rightarrow$ after $\leq |f|$ iterations, the max-flow is found.

  The theorem follows because each iteration (DFS) takes $O(m)$!

- Caveat: for large $|f|$, this can be very slow.

Ford Fulkerson additional notes!

- *Shortest path* variant of Ford-Fulkerson runs in time $O(m^2 n)$
- *Fattest path* variant of Ford-Fulkerson runs in time $O(m^2 \log n \log|f|)$
- If all the capacities are integers, then the flow on any edge in any max flow is also integral.
  - When we update flows in Ford-Fulkerson, we're only ever adding or subtracting integers.
  - Since we started with 0 (an integer), everything stays integral.