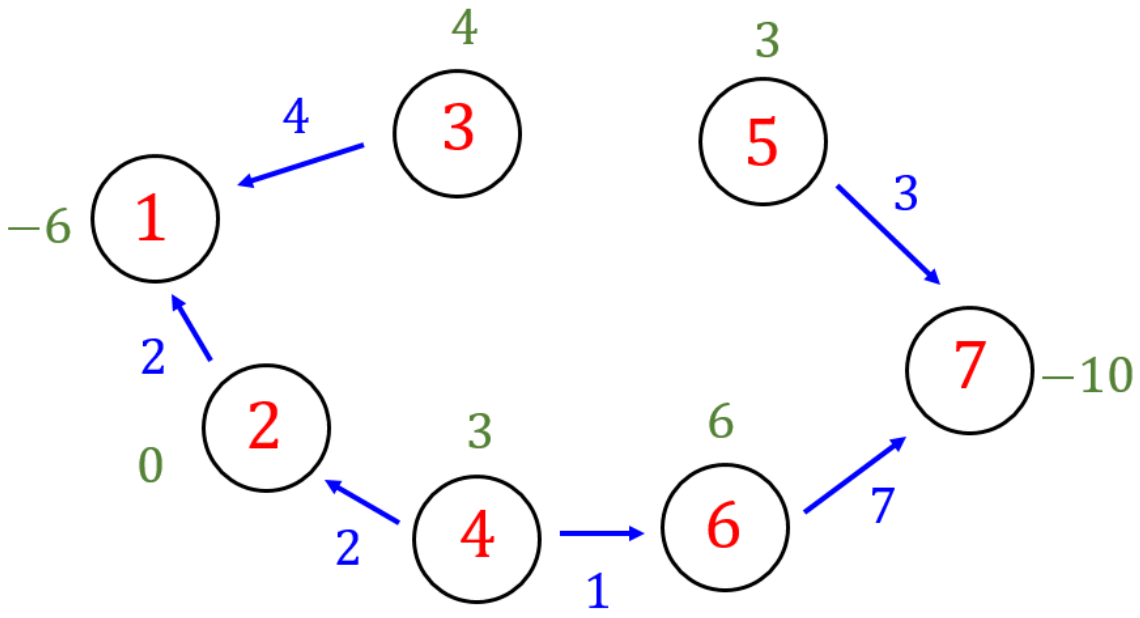
LECTURE 16: MAX FLOX/MIN CUT

1. Network Simplex Algorithm (continued)

Recall: How to implement the simplex algorithm on network problems

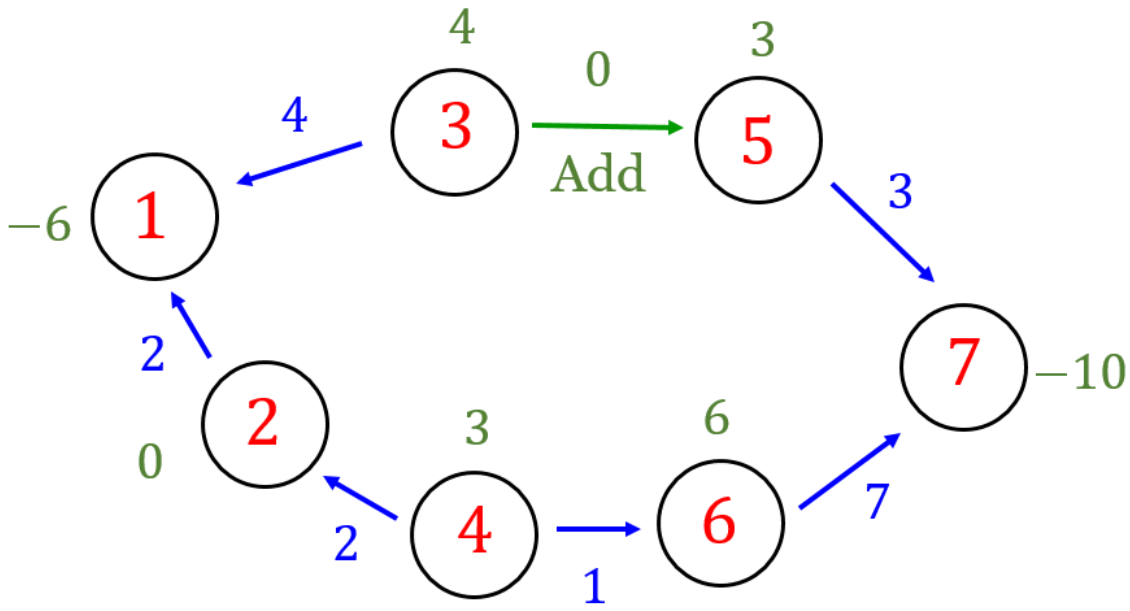
Example 1:

Suppose we have a graph where one tree solution is

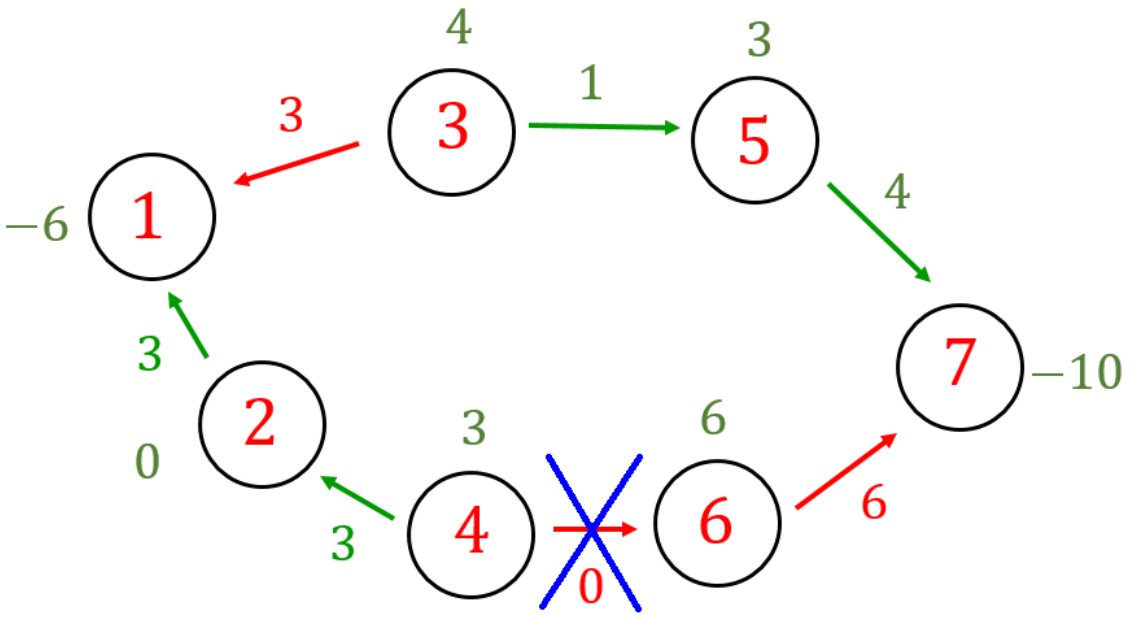


Add an edge, say $3 \to 5$ with value 0 (assume part of the original graph)

Date: Tuesday, November 1, 2022.



Increase the value on $3 \to 5$ until one edge becomes 0, here $4 \to 6$ That is the edge we want to remove.



Note: We increased the value of $3 \to 5$ by $x^\star$ where

Definition:

$$x^\star = \min x_{ij}$$

i, j ranges over the edges $i \to j$ in the *opposite* direction of $3 \to 5$

In this case $x_{67} = 7, x_{46} = 1, x_{31} = 4$ and so $x^\star = \min\{7, 1, 4\} = 1$

Change in z

We added $x^\star$ to each edge in the same direction as $3 \to 5$ (green) and removed $x^\star$ from each edge in the opposite direction (red), so the overall change in z is

$x^\star \times$(Cost on edges in the same direction as $3 \to 5 -$ Cost in opposite direction)

The number in parentheses is called the **reduced cost**:

Definition: (Reduced cost/Change in cost)

$$\overline{c_{35}} = \sum_{ij \text{ same}} c_{ij} - \sum_{ij \text{ opposite}} c_{ij}$$

In this case we have

$$\overline{c_{35}} = (c_{35} + c_{57} + c_{42} + c_{21}) - (c_{67} + c_{46} + c_{31})$$

- If $\overline{c_{35}} < 0$, then $3 \to 5$ is a better edge and you continue with this new tree.
- If $\overline{c_{35}} \geq 0$, then it's not a better tree and you continue with a different edge.

Optimality Test:

If $\overline{c_{ij}} \geq 0$ for all edges not in the tree, then we have an optimal tree and we stop.

Note: This is also what you check at the very beginning when checking for optimality

Important Note about Cycles: Suppose you add the edge $5 \to 6$. Then you get a mini-cycle $5 - 6 - 7$ in the tree. In this case you **only** focus on that cycle. This means you calculate the reduced cost *of that cycle*, which is $c_{56} + c_{67} - c_{57}$, and you increase the weight of $5 \to 6$ until an opposite edge in the cycle $5 - 6 - 7$ becomes 0, so in this case $x^\star = 3$ and $5 \to 7$ would get removed. You ignore the other edges here.

2. Max Flow

Let's now revisit and solve max flows

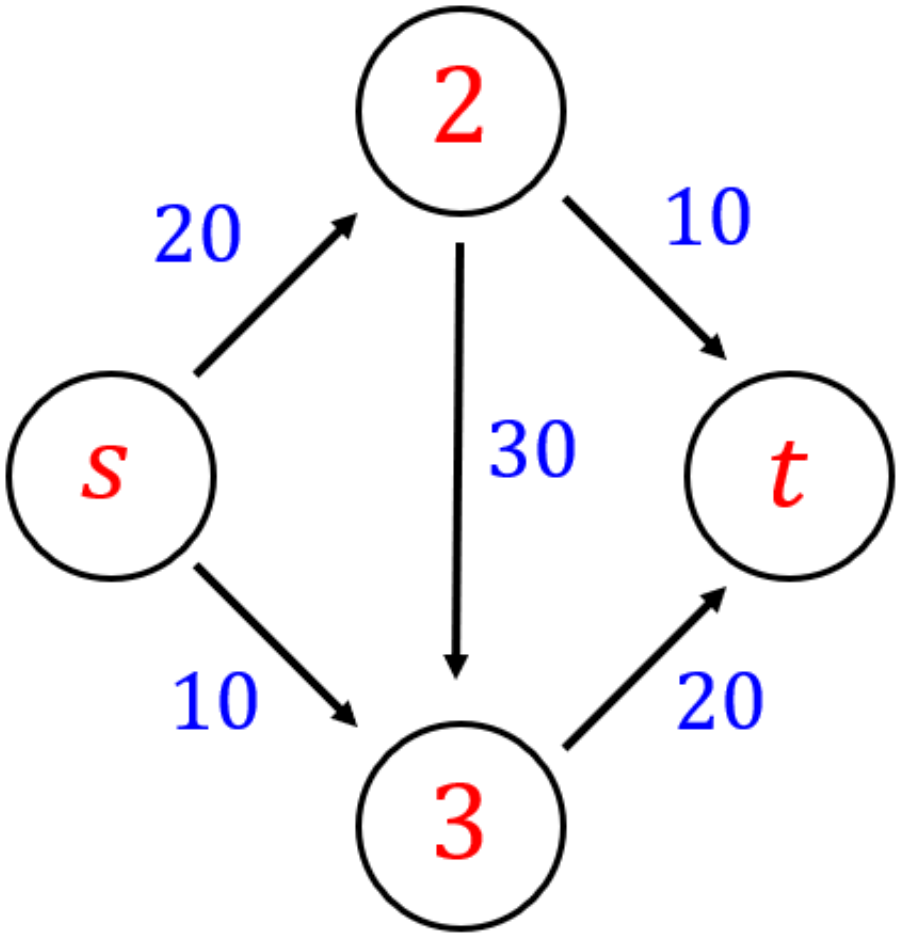
Example 2:

Consider the following network

Here the cost is 1 and the number on each edge is the max capacity

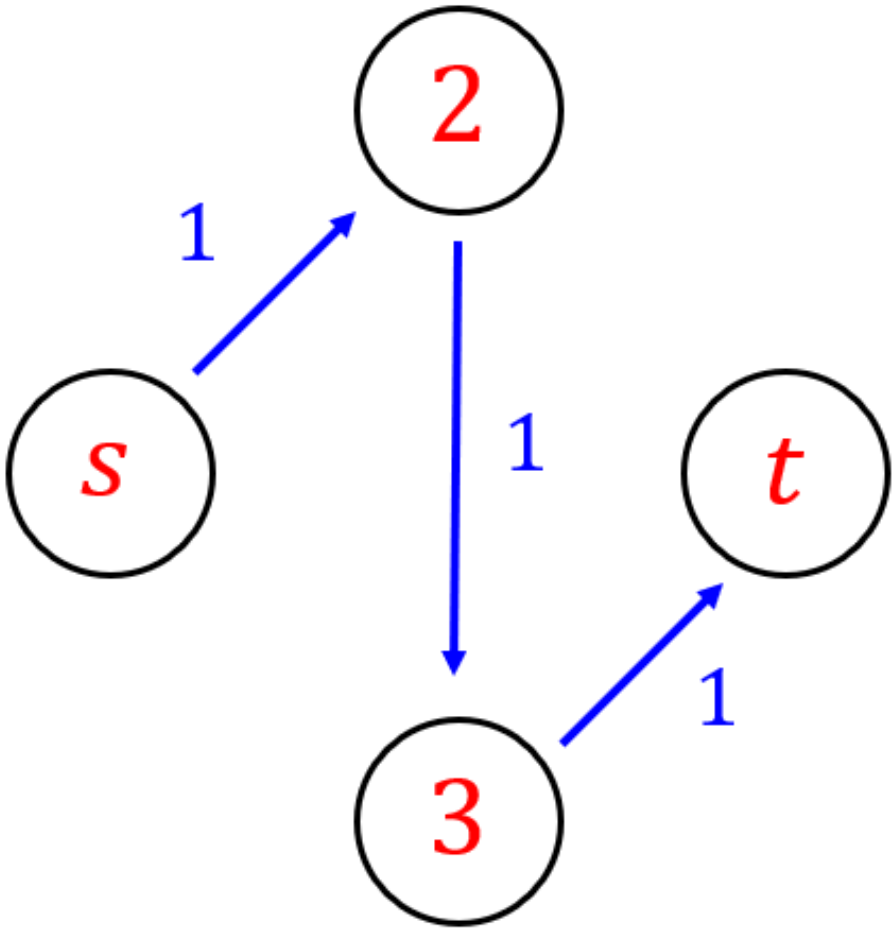
Goal: Maximize the supply at s

That is, carry as many products from s to t, while satisfying the capacity constraints.



STEP 1: Initial Tree

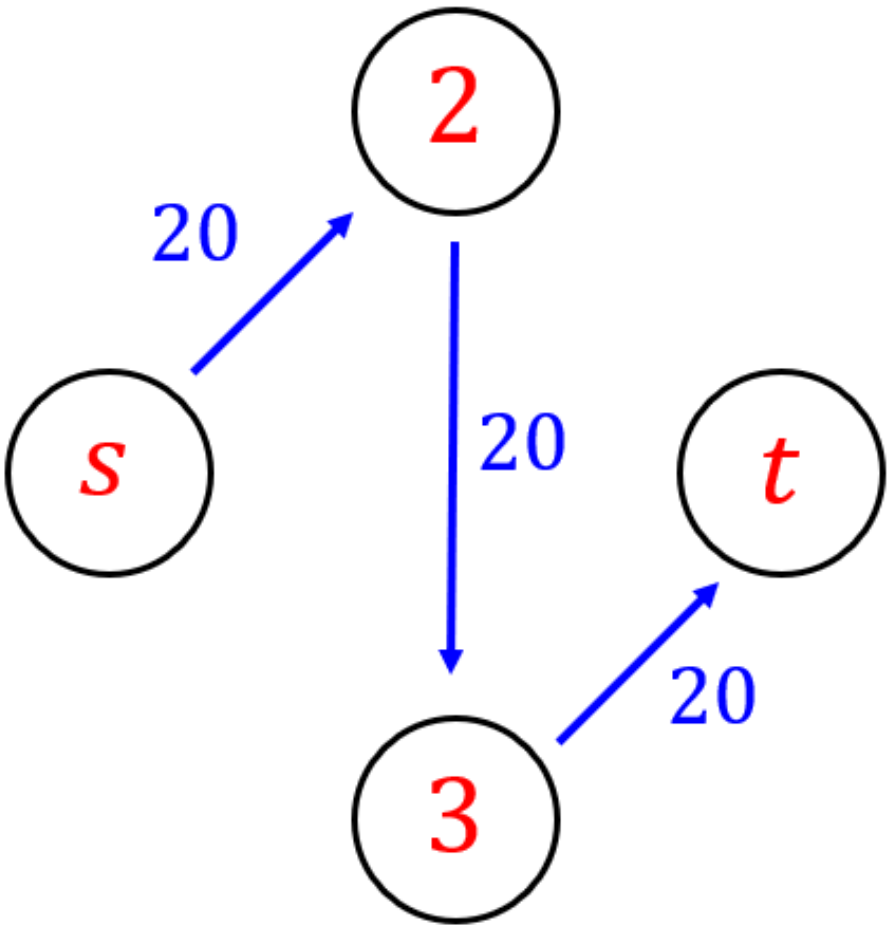
Start with any path from s to t and put weight 1 on each edge:



(doesn't have to go through all the vertices)

Note: Can do better, can assign the weight on each edge to be $\min\{20, 30, 20\} = 20$, the smallest possible weight on the path.

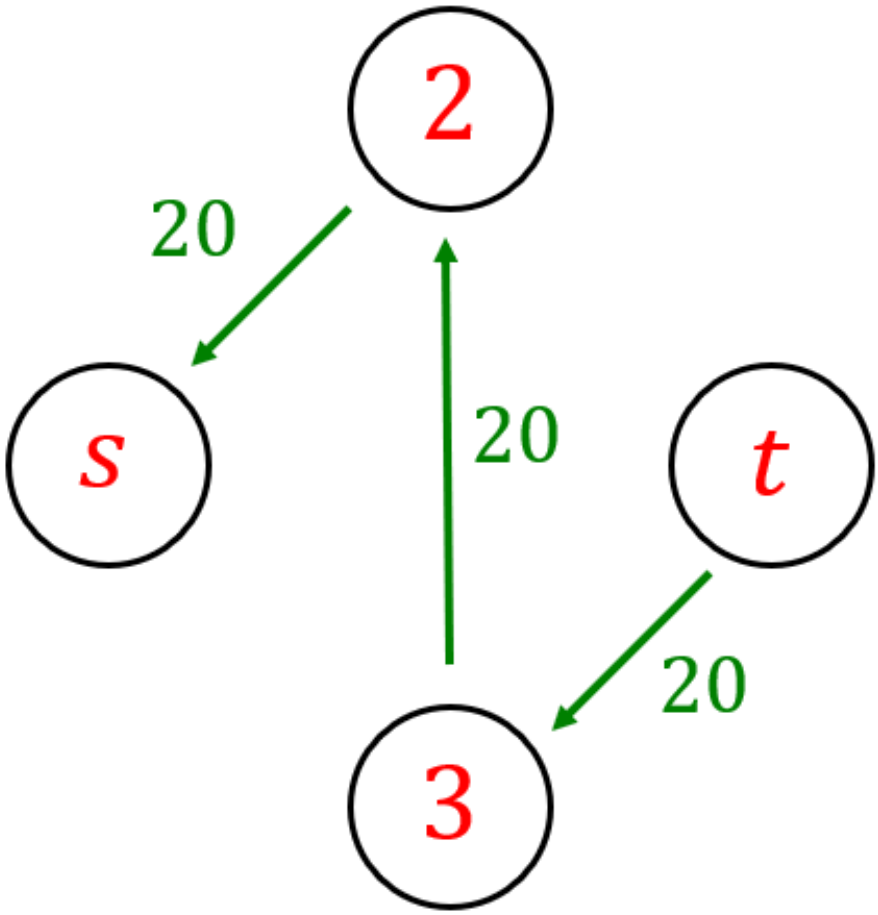
Initial Tree:



STEP 2: Residual Graph

Think of this as a shadow/helper graph, one that will help us complete the picture

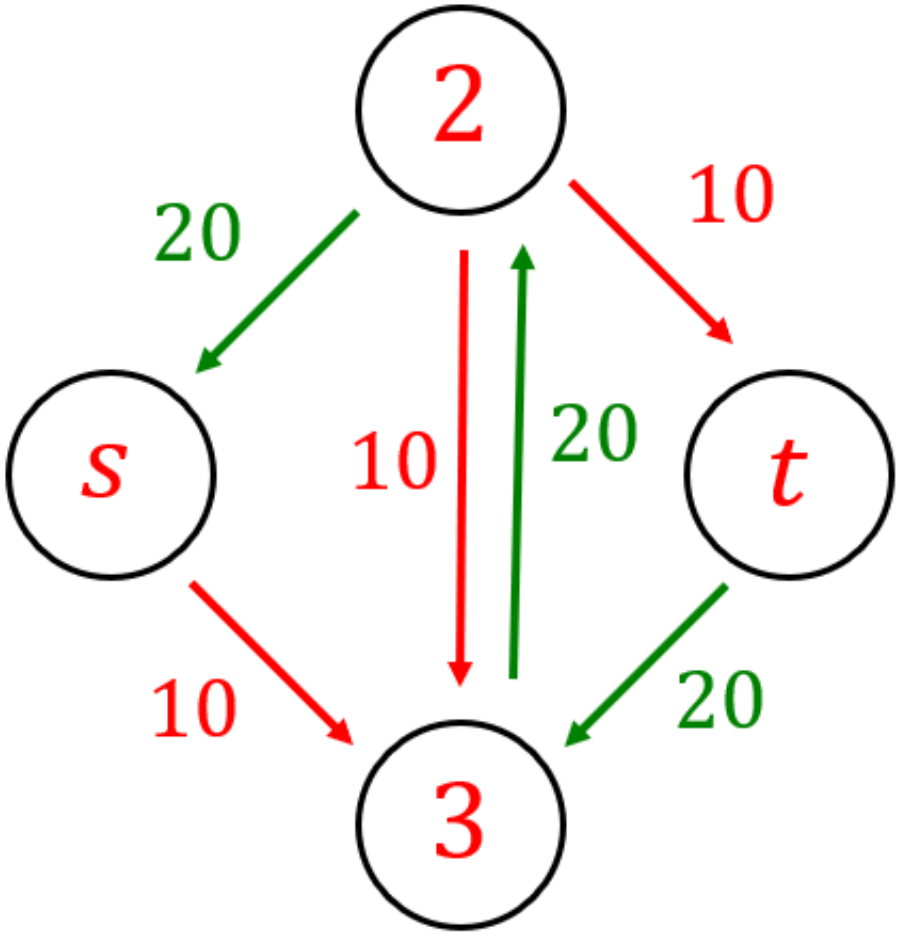
Rule 1: Reverse all the (non-zero) arrows on the path



Rule 2: For the arrows that are not maxed out, add an arrow in the same direction but with the remainder as weight.

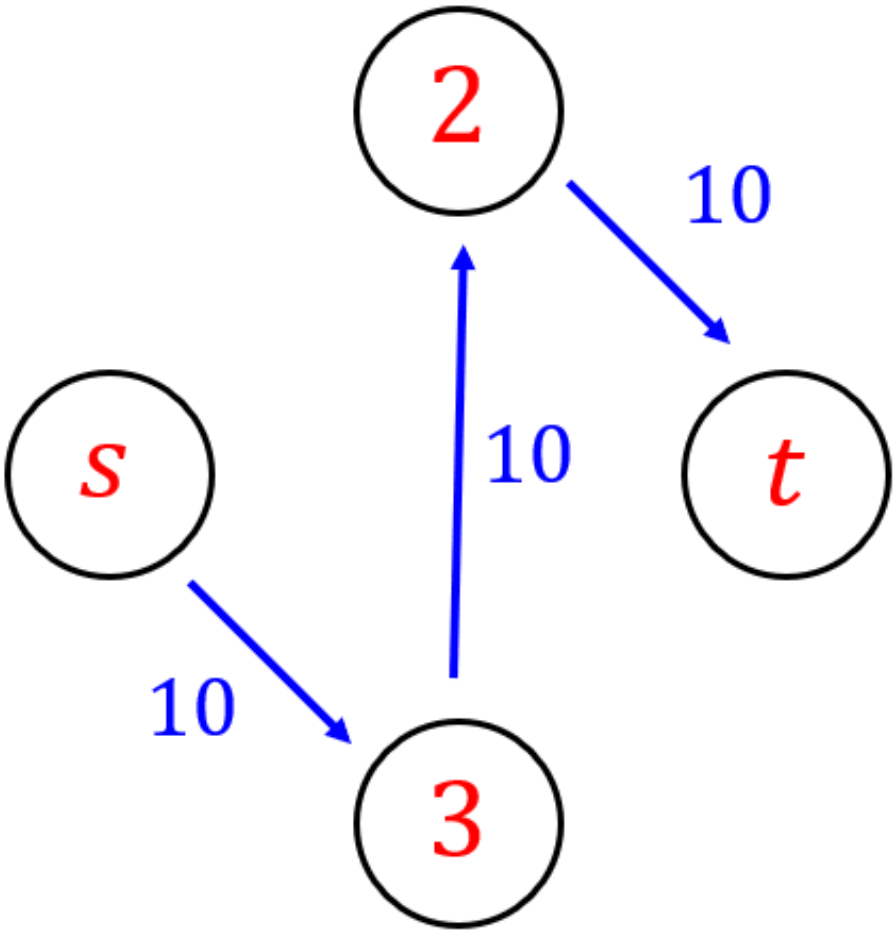
$2 \to 3$ is not maxed out since the capacity on that edge is 30 but we only used 20 so we add an arrow $2 \to 3$ with weight $30 - 20 = 10$

We also add arrows $s \to 3$ and $2 \to t$ with weights $10 - 0 = 10$ since we used 0 but the max weight is 10


2
20
10
s
10
20
t
10
20
3

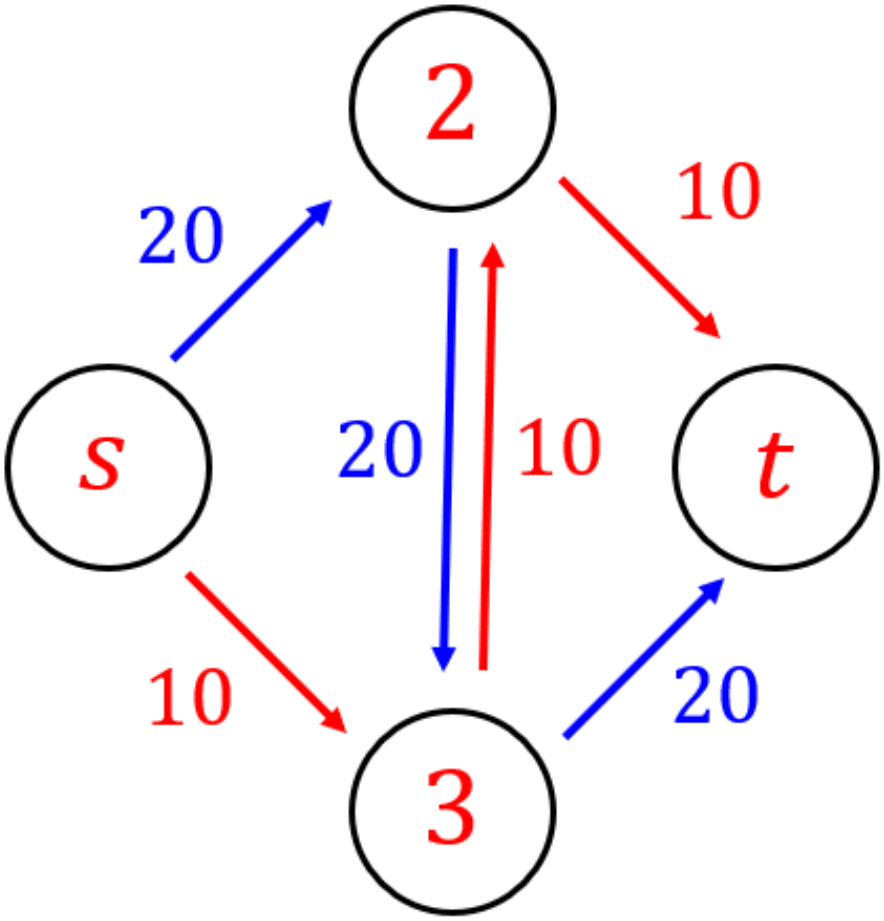
STEP 3: Do **STEP 1** but with the residual graph (find a path and assign minimal weight)

Here a path is $s \to 3 \to 2 \to t$ and you assign weight 10



STEP 4: Augmenting the graph

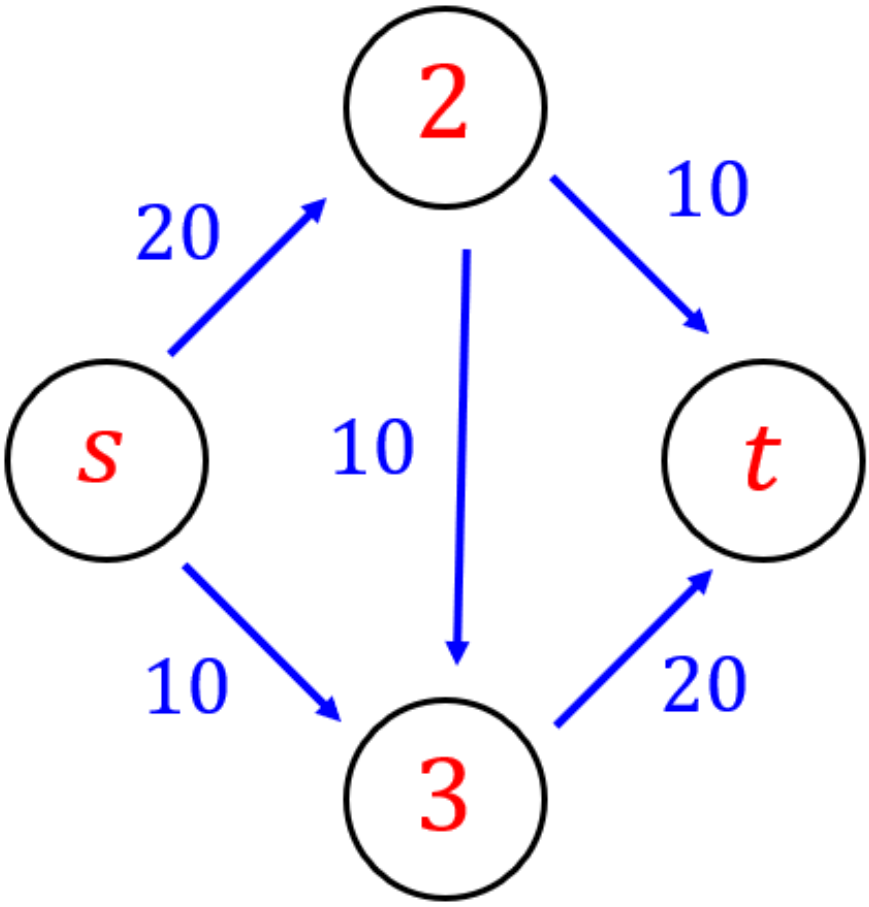
Add the graph in **STEP 3** (red) to your graph from **STEP 1** (blue)



Note: Here you have to think of $3 \to 2$ as $2 \to 3$ but with weight -10, so the total weight is $20 + (-10) = 10$.

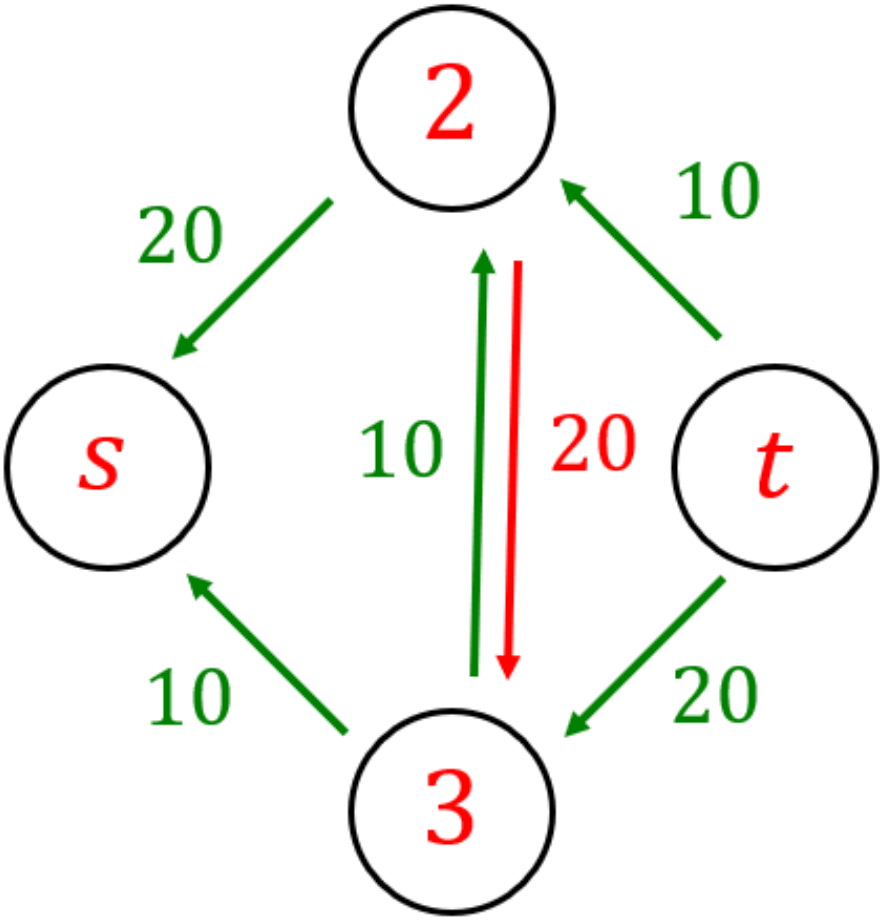
Note: If you had two edges $s \to 2$ in the same direction but with values 2 and 3, this would become one edge $s \to 2$ with value $2 + 3 = 5$

Hence our updated graph looks like:



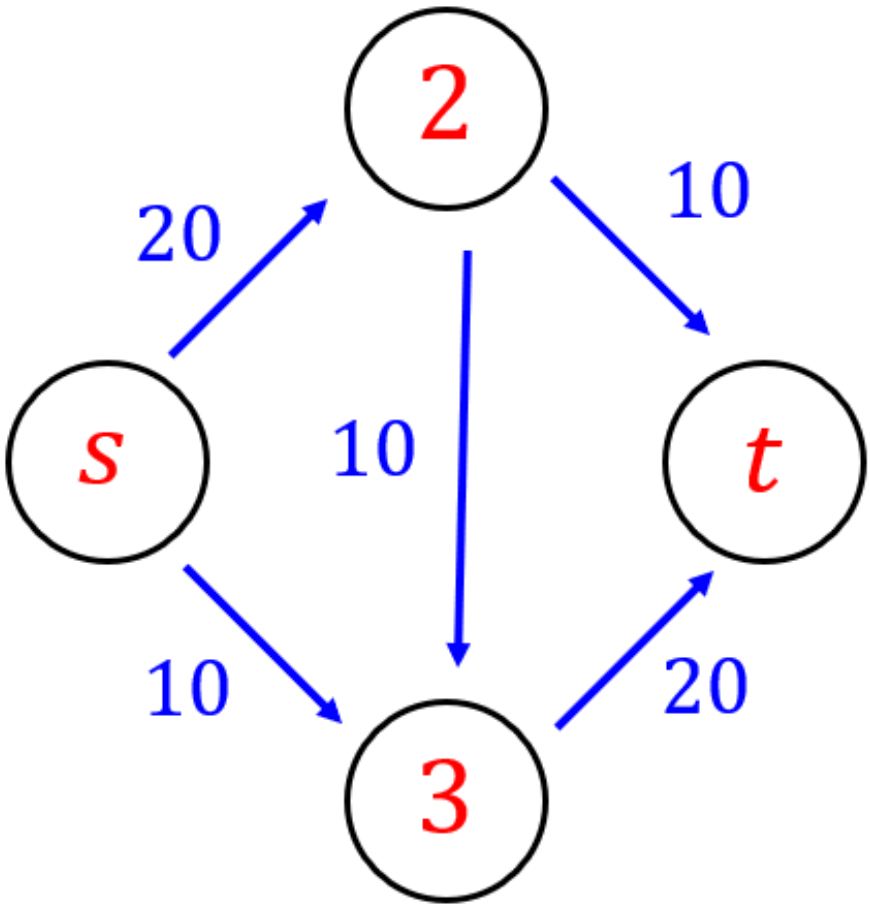
STEPS 5+ Rinse-and-Repeat

Find the residual graph of this, find a path, and add it to your graph



We're stuck, there is no path from s to t, at which point we **STOP**

Optimal Solution:

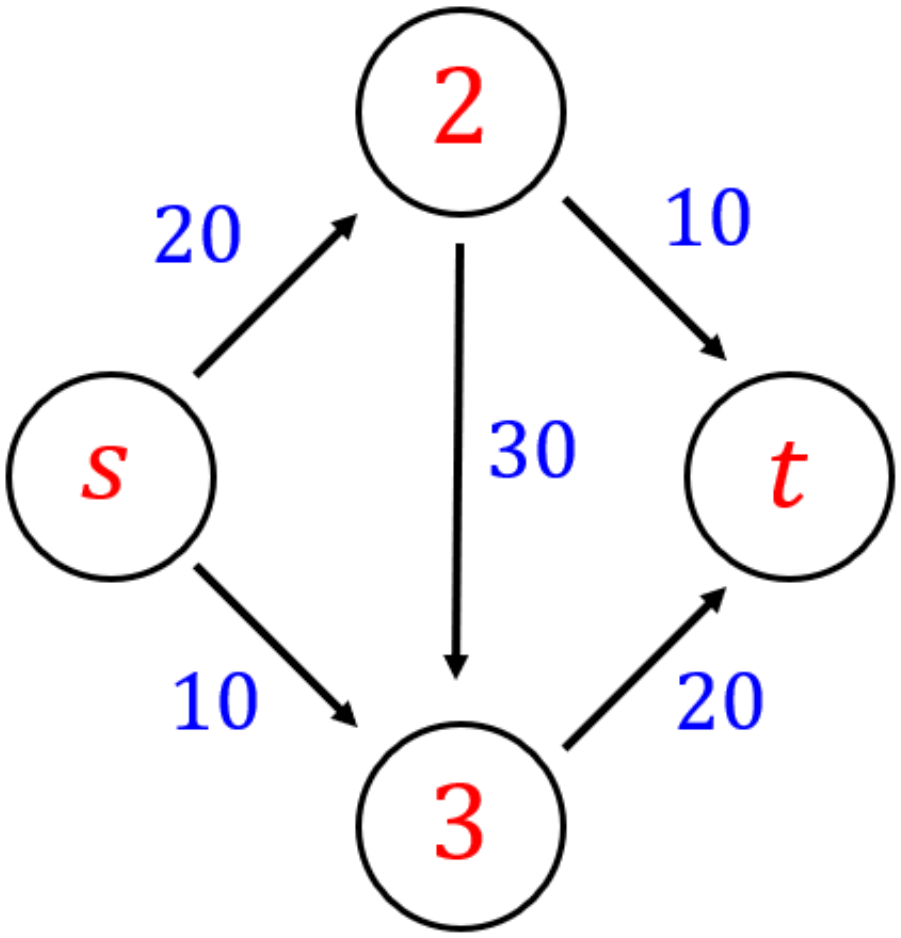


Max Flow: (largest amount we can produce at s) $20 + 10 = 30$

3. Min Cut

The dual way of thinking about the previous problem!

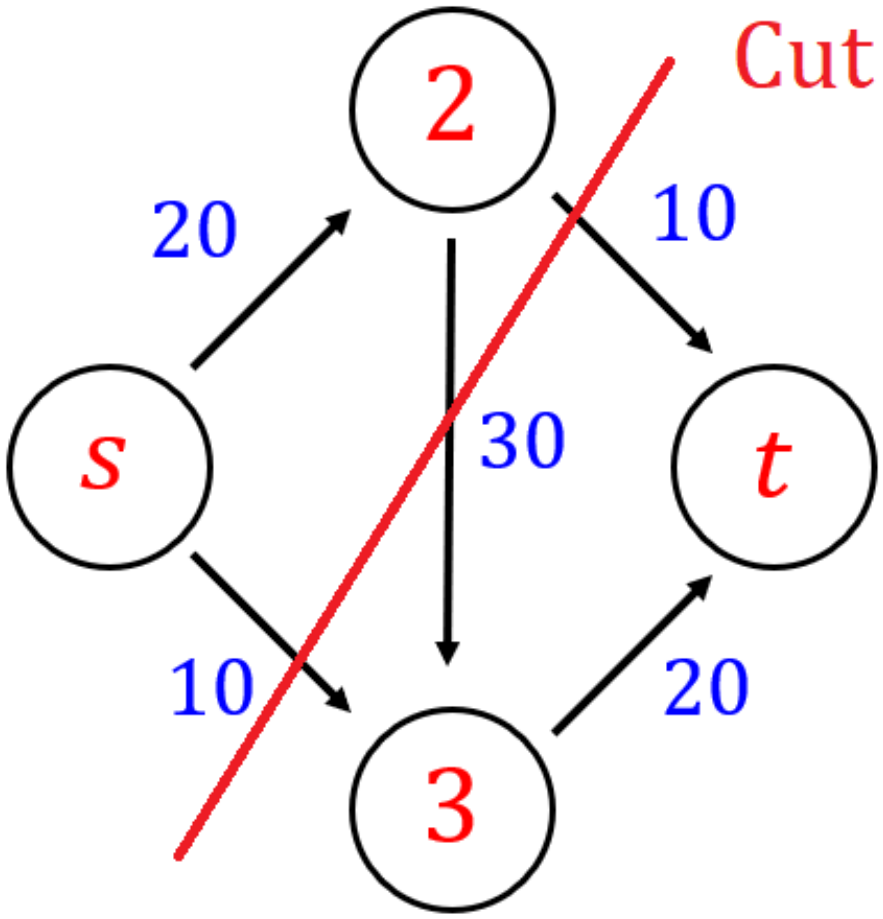
Back to original problem:



Definition:

A **cut** is any curve through the graph that separates s and t

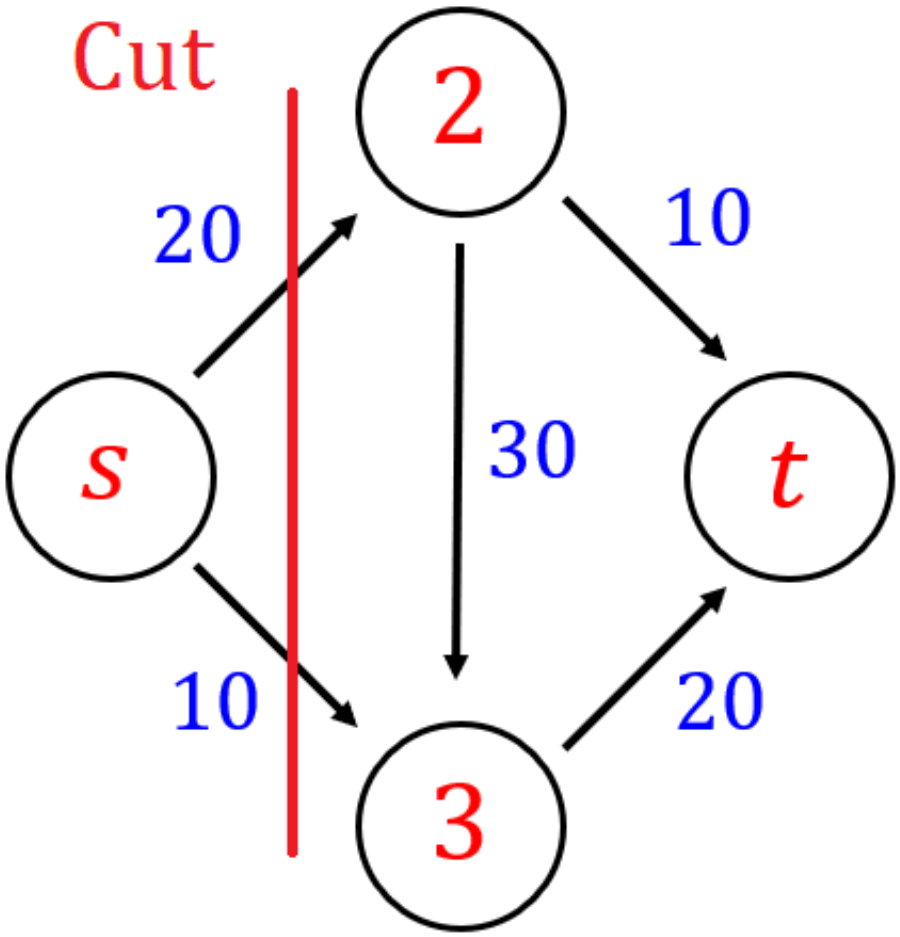
Think of it as a barrier or a wall.



The largest amount we can get across the cut above is $10+30+10 = 50$

What does that have to do with max flow? Any path from s to t *has* to go through a cut, so the max flow (max amount that we can ship) has to be *at most* 50

Can we get a tighter bound? Yes! Consider the following cut:



The largest amount we can get across that cut is $20 + 10 = 30$

So really what we want to do is to **minimize** this, we want to minimize the largest amount we can get across all possible cuts

Min Cut Problem:

$$\min_{C} \quad \text{largest amount across } C$$

(where C ranges over all cuts)

Max Flow/Min Cut Theorem:

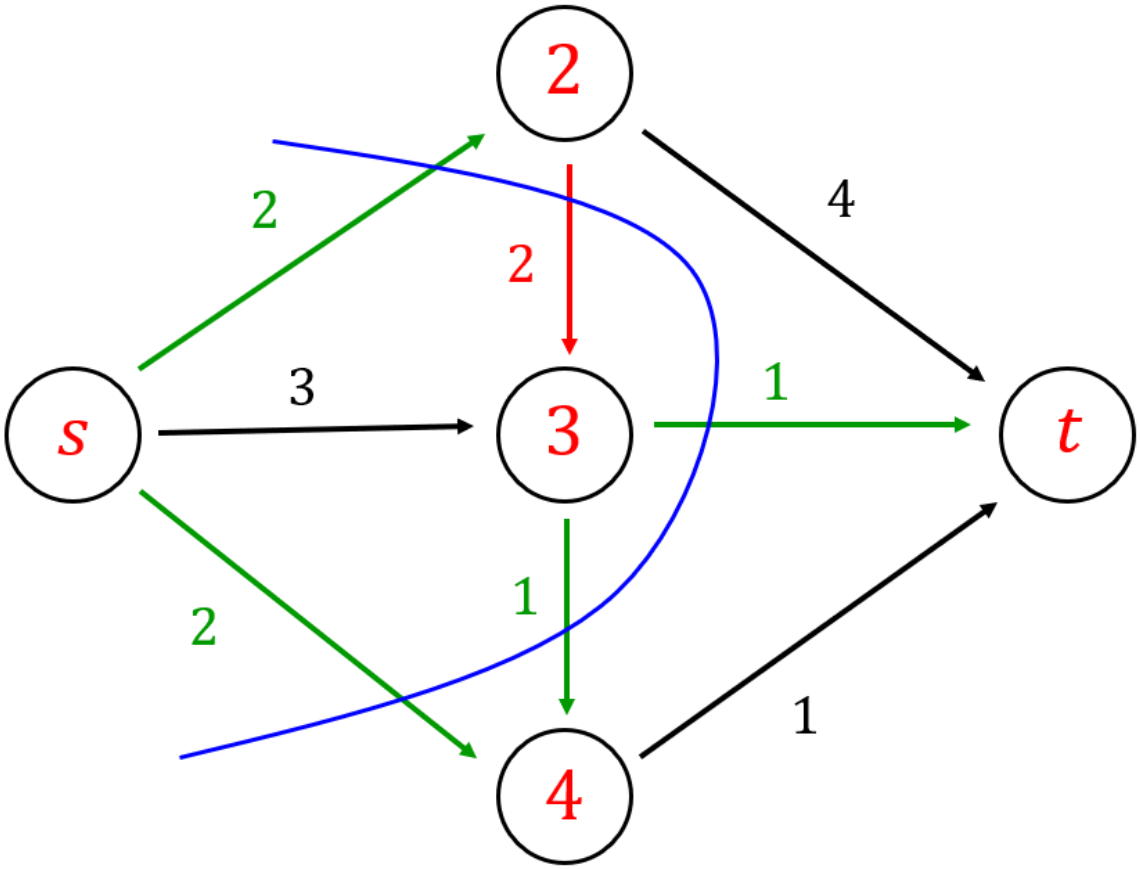
$$\text{Max Flow} \;=\; \text{Min Cut}$$

In the example above, can check that the min of all the cuts is indeed 30 which is also the value of the max flow that we found.

The only issue is that this procedure doesn't tell us what the optimal graph looks like, just how much to produce at s

Note: The min cut problem is indeed dual to max flow, if you remember the motivation in the Dual LP lecture, where we concluded that, playing around with the constraints, we have $z \leq 200$, and playing around some more, we got $z \leq 190$ and we concluded that we had to minimize this number 190

Note: A cut can be any curve through the graph that separates s and t, like the blue curve in the example below:



This separates the graph into two regions, one that contains s and one that contains t.

Moreover, when calculating the value of the cut, you only consider edges going *out* of the $s-$region. So for instance here you don't count the edge $2 \to 3$, and the value of that cut here is $2 + 1 + 1 + 2 = 6$.