LECTURE (October 21st)

Maximum Flows & Minimum Cuts

Today we are going to look at flows, which might be a more familiar territory

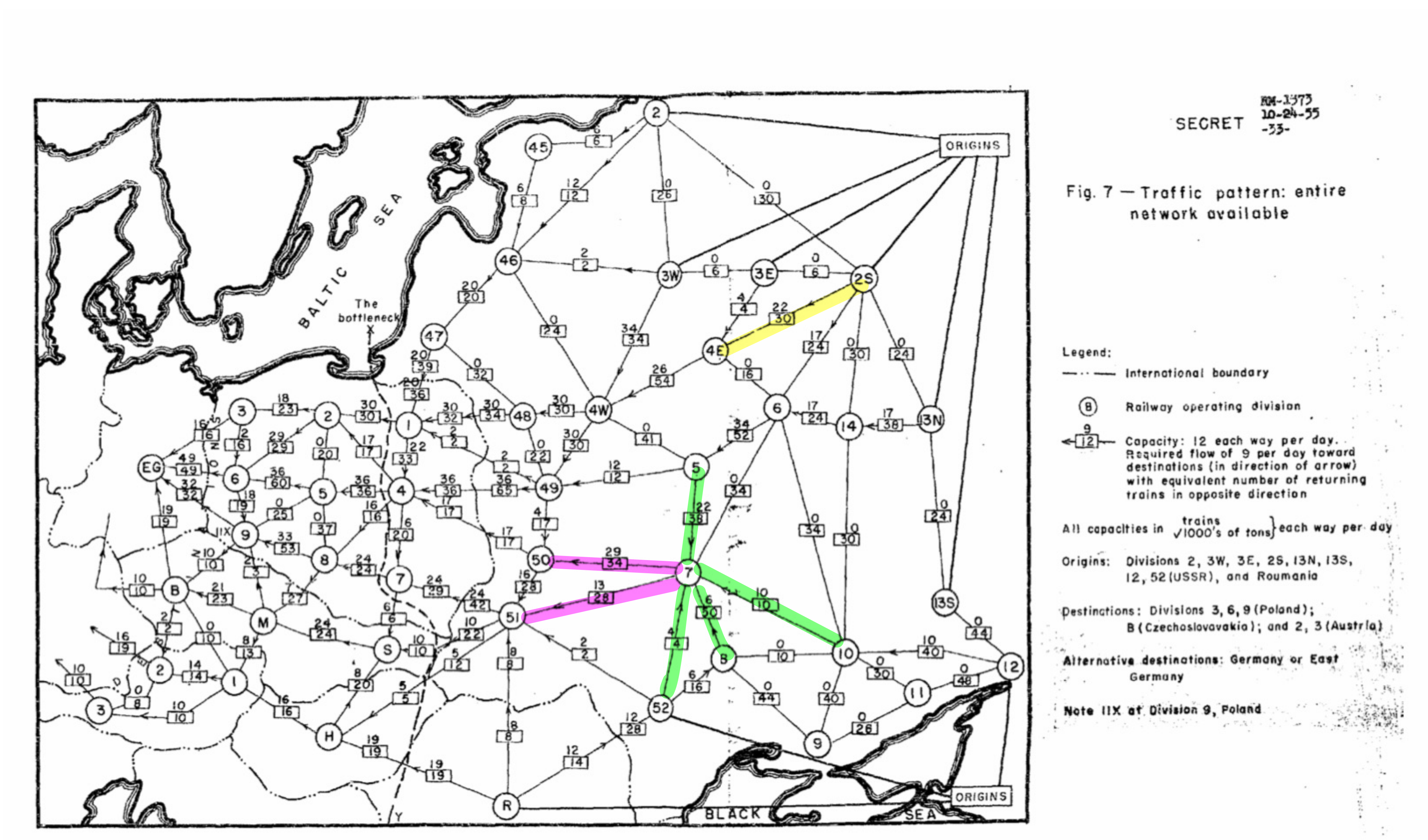
Given a graph with some data associated to it, we want to compute some structure within that graph

You might have seen related examples in previous courses like CS225, where you saw minimum spanning trees & Dijkstra's algorithm, CS374 where you saw all pairs shortest path algorithm. Those structures that you are computing — shortest path, minimum spanning tree, etc. — are subgraphs of your input graph

In particular, you want to pick out a subgraph of your graph that satisfy some optimality properties

Today, we are going to look at other kinds of optimal structure

Before we look at the definition, let us take a detour into history of the problem



This is a map of Eastern Europe after the second world war when cold war was looming

The circles represent cities & the labels are just identifier
The numbers on the box between two cities represent the number of trains that go between the cities daily, think of it as the capacity of the rail line

The numbers outside the boxes and the arrow represents a schedule to use some sort of material which is mined at the box labeled origins

So, we want to ship material from Moscow to East Berlin
Instead of using the entire capacity, we can send fewer trains on a single track

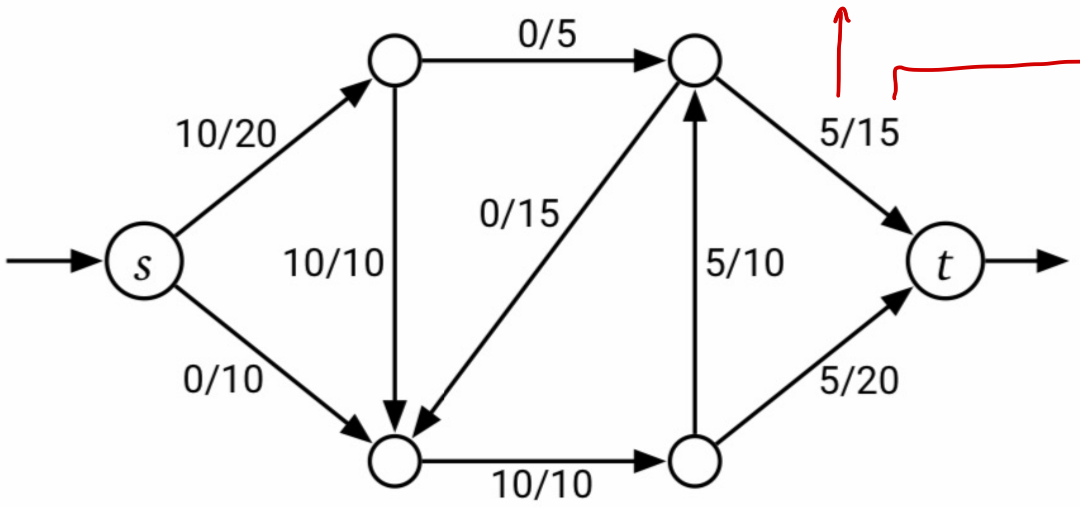
If you look at any city that is not Moscow or East Berlin, then you will notice that the amount of stuff going into the city is the same as the amount of stuff going out, since it does not make sense to have extra stuff flowing into intermediate city and by definition the material is only produced at the origin, so it does not make sense for more stuff to leave the city than what came in

This is an example of the maximum flow problem

The input is a directed graph with two special vertices — source and sink

flow on the edge

capacity



• What is the value of this flow?
• Is this a maximum flow?

Each edge has a number associated to it called the capacity and you want to compute a second number for every edge called the flow value that satisfies the conservation constraint at all intermediate vertices, i.e., the flow coming in equals the flow coming out

Maximum Flow Problem Given $G = (V, E)$ directed
capacity function $c : E \to \mathbb{R}_{\geq 0}$
two vertices source s & target t

Want Flow which is a function $f : E \to \mathbb{R}_{\geq 0}$ satisfying

• Any function satisfying conservation constraints $\boxed{1}$ is called a flow

$\boxed{1}$ $\sum_u f(u \to v) = \sum_w f(v \to w) \quad \forall v \neq s, t$

$\boxed{2}$ $0 \leq f(e) \leq c(e)$

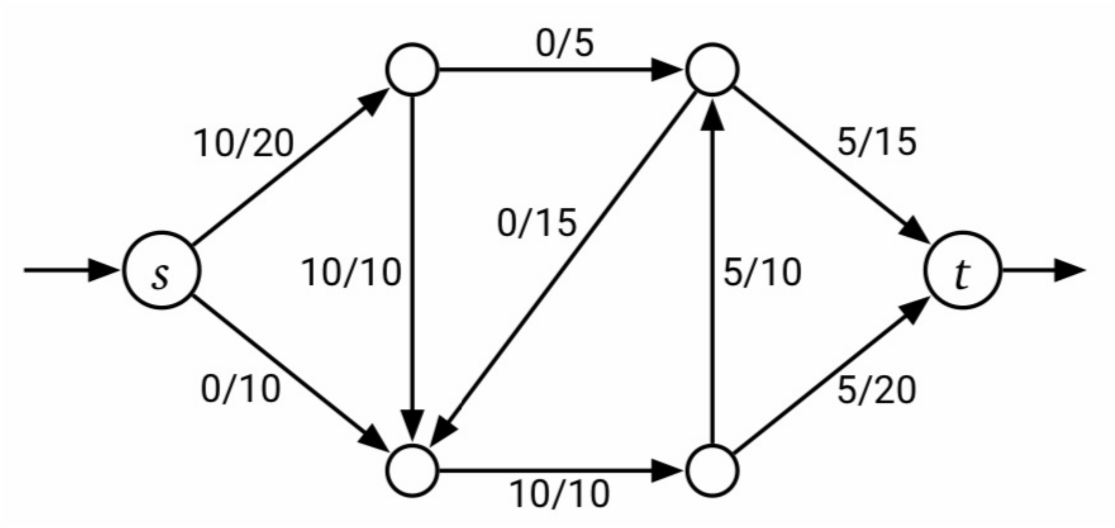
• Any flow satisfying capacity constraints $\boxed{2}$ is called feasible

$\boxed{3}$ Value of flow $|f| = \sum_w f(s \to w) - \sum_u f(u \to s)$

• Note that by conservation
$|f| = \sum_u f(u \to t) - \sum_w f(t \to w)$

We want the flow with the maximum value

One useful analogy is to think of the edges as pipes and the capacity as the capacity of the pipe before it explodes

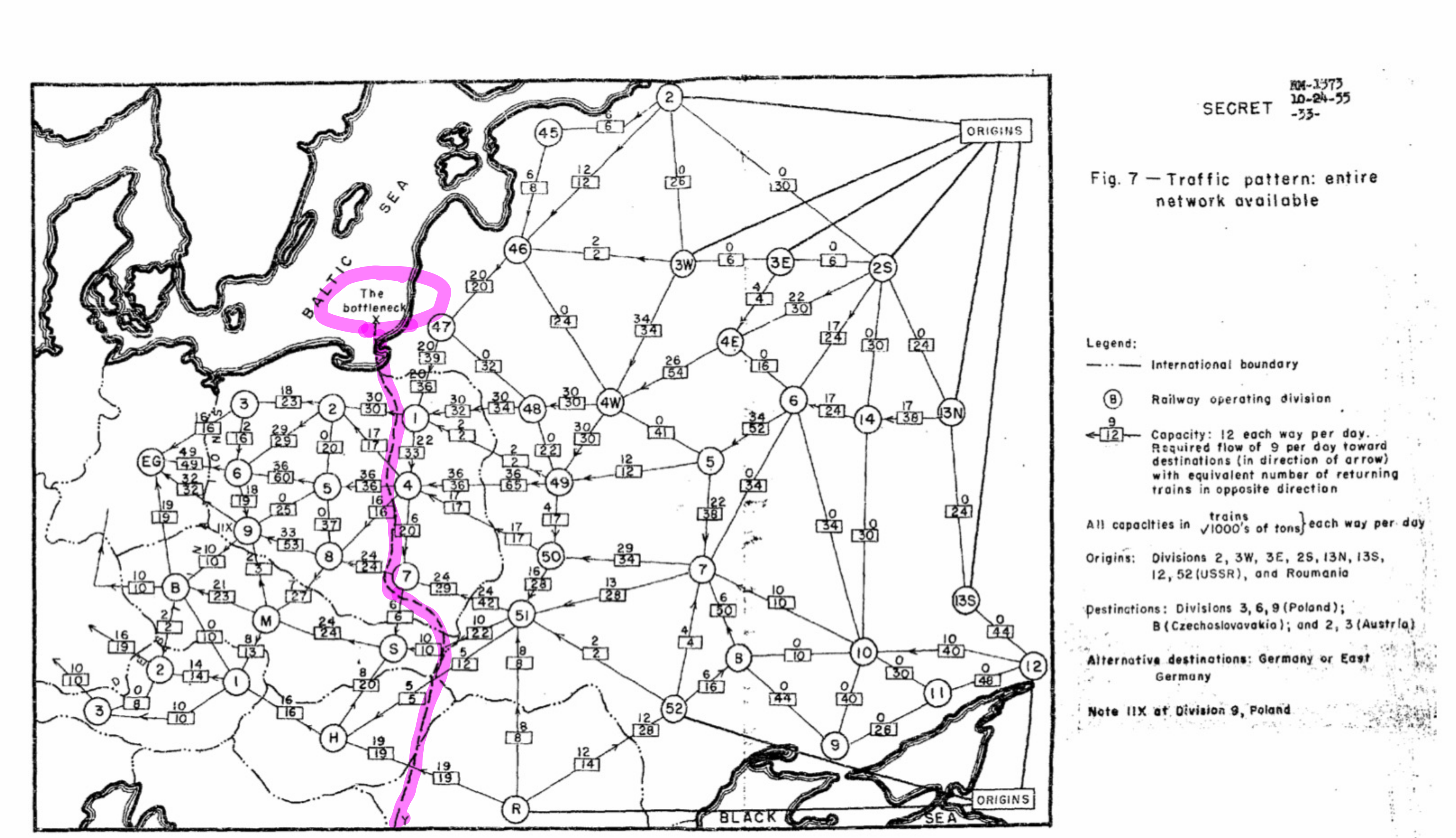


• What is the value of this flow?

• Is this a maximum flow?

We will see shortly why this is not a maximum flow & how to get to a maximum flow from here

Before we do that, let's introduce the evil twin of the maximum flow problem called the minimum cut problem



This map was made by RAND corporation in the 50s — a lot of algorithm research originated there in the 1950s

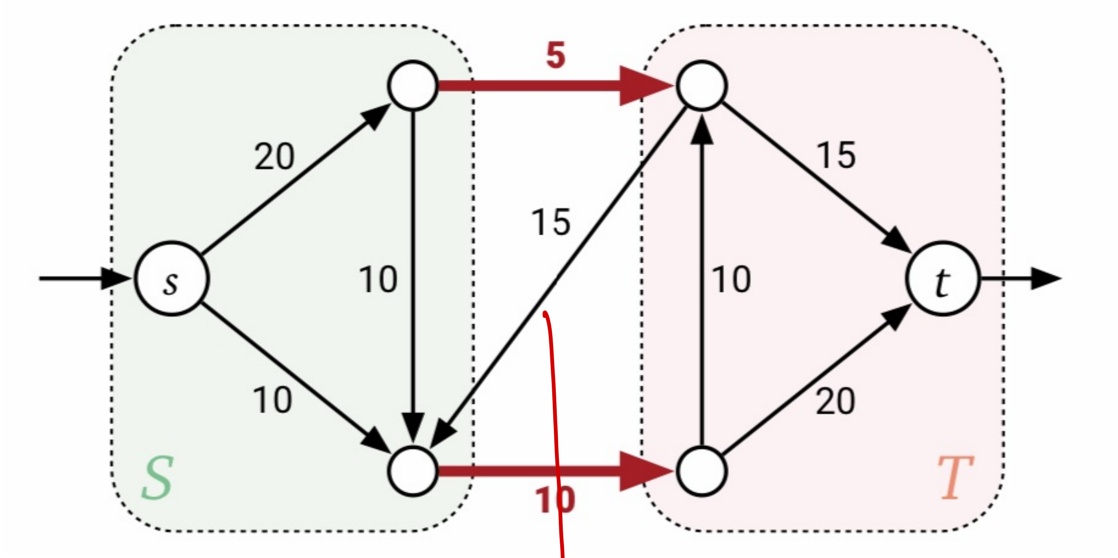
This map was classified until 1999, when an optimization researcher Lex Schriver wrote to the US government to declassify it

The bottleneck is the smallest cost of destroying all the rail lines to disconnect Moscow from East Berlin — cost equals the capacity of the rail line

This is called the minimum cut problem. The input is exactly the same with a source s & target t but now we are trying to separate the source & target, i.e., divide the vertices in two parts where one contains s & the other contains t. Such a partition is called a cut.

Cut Partition $V = S \cup T$ where $s \in T$ & $t \in T$ & $S \cap T = \emptyset$

Capacity of a cut $|(S,T)| = \sum_{\substack{u \in S \\ v \in T}} c(u \to v)$ ↳ We only care about flow from s to t not the other way around



This edge is irrelevant

What is the capacity of the cut $(S \cup T)$?

We want the cut with the smallest capacity, i.e., the smallest cost to disconnect s from t

The max-flow min-cut theorem, whose proof we will see in today's lecture says that in a given graph

Max-flow value = Min-cut capacity

Max-flow Min-cut Theorem

$$\max_{\text{flow } f} |f| = \min_{S \cup T} |(S,T)|$$

The proof of this theorem will also give us an algorithm to compute both the maximum flow & the minimum cut

Let's see the easy direction of the proof first $\max |f| \le \min |(S,T)|$

Pick any feasible flow f and any cut (S,T)

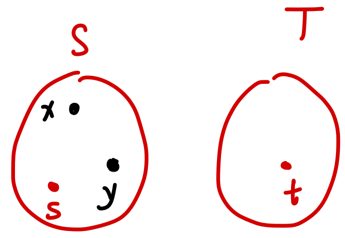
Value of f is $|f| = \sum_w f(s \to w) - \sum_u f(u \to s)$

Since all vertices except s,t have the same of flow going in as coming out, we have

$$\forall\, v \neq s,t \qquad \sum_{w} f(v\to w) - \sum_{u} f(u \to v) = 0$$

Thus, adding it to the quantity above

$$|f| = \sum_{w} f(s\to w) - \sum_{u} f(u\to s)$$
$$+ \sum_{v \in S\setminus\{s\}} \left(\sum_{w} f(v\to w) - \sum_{u} f(u\to v) \right)$$



$$= \sum_{v\in S} \sum_{w} f(v\to w) - \sum_{v\in S} \sum_{u} f(u\to v)$$

Consider an edge $x\to y$, s.t. both x & y are in S

Note that $f(x\to y)$ appears in the first sum when $v=x$ & $w=y$

& $-f(x\to y)$ appears in the second sum when $u=x$ & $v=y$

$$= \sum_{v\in S} \sum_{w\notin S} f(v\to w) - \sum_{v\in S} \sum_{u\notin S} f(u\to v)$$

$$= \underbrace{\sum_{v\in S} \sum_{w\in T} f(v\to w)}_{\text{flow from } S \text{ to } T} - \underbrace{\sum_{v\in S} \sum_{u\in T} f(u\to v)}_{\geq 0} \quad \hookrightarrow \text{flow from } T \text{ to } S$$

$$\leq \sum_{v\in S} \sum_{w\in T} \underbrace{f(v\to w)}_{\leq c(v\to w)} \qquad \leq \ \|(S,T)\|$$

Therefore, $|f| \leq \|(S,T)\|$ for any flow f & cut $S \cup T$

$$\Rightarrow \max |f| \leq \min \|(S,T)\|$$

We have used all the properties of the flow
- value of the flow
- conservation
- non-negativity
- feasibility

But one of the thing this implies that if we find a flow & a cut that have the same value, then the flow must be a maximum flow & cut must be the minimum cut and all the inequalities must be tight

If $|f| = |(S,T)|$, then

- f is a max-flow
- $S \cup T$ is a min-cut

- $f(u \to v) = c(u \to v)$ for all $u \in S, v \in T$
- $f(u \to v) = 0$ for all $u \in T, v \in S$

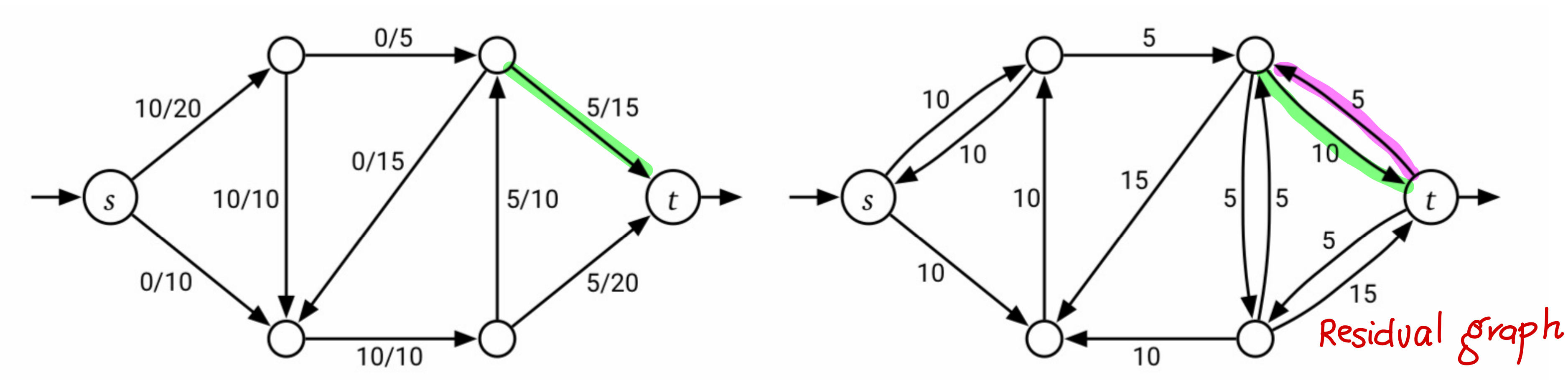
Let's see the proof of the other direction

$$\max |f| \geq \min |(S,T)|$$

The proof will be easier if we assume there is at most one edge between any two vertices — this is easy to handle [Why?]

Pick your favorite flow f, we define a new flow problem using residual capacities & residual graphs

These capture how much of the capacity is not being used

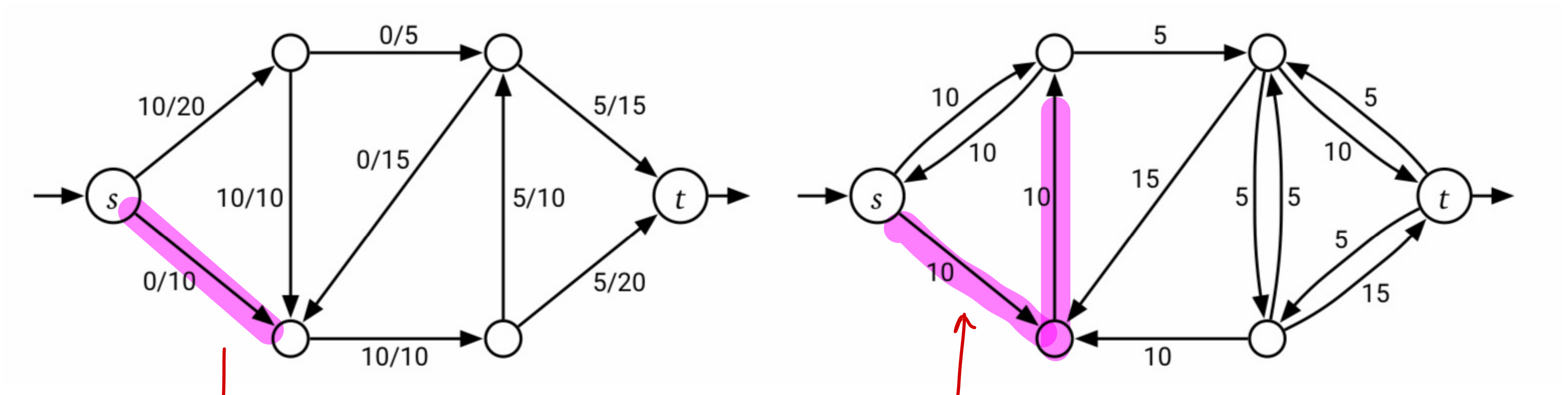


Residual graph

Define residual capacity for flow f by

$$c_f(u \to v) = \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \in E \\ f(v \to u) & \text{if } v \to u \in E \\ 0 & \text{o/w} \end{cases}$$

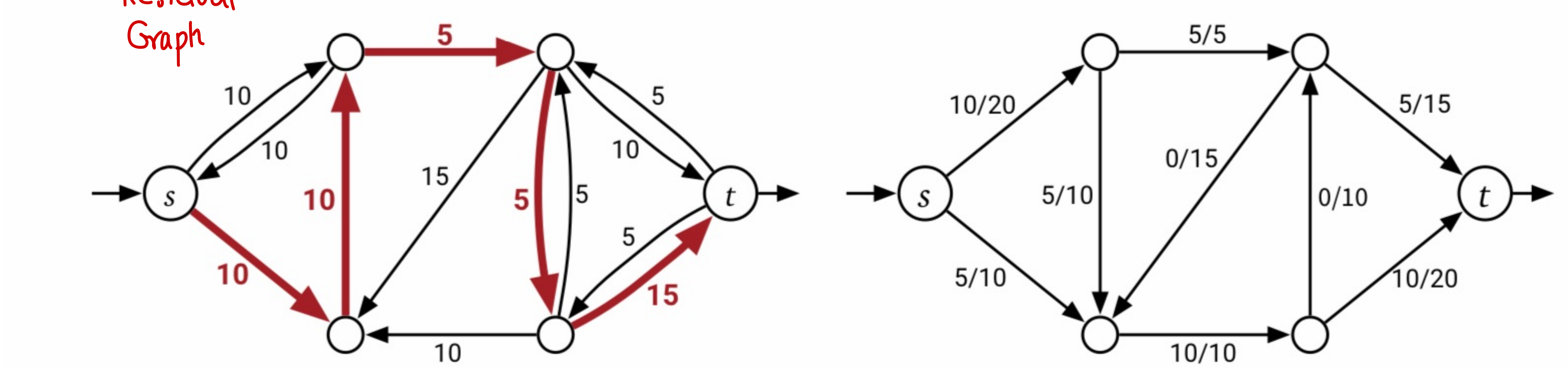
Residual capacities are always non-negative [Why?]



There is still capacity left on this edge called an augmenting path

Case 1 : There is a path from s to t in the residual graph

Residual Graph



Consider any augmenting path, the maximum flow we can push through the path is the min residual capacity of any edge on the path

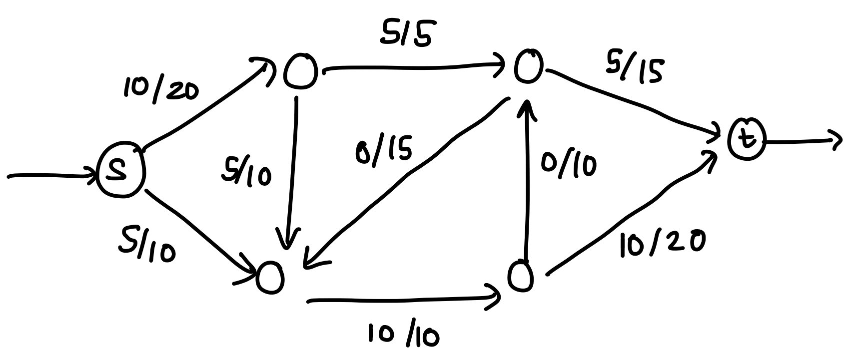
Let c_{min} = min-capacity of edges along any path p from s to t

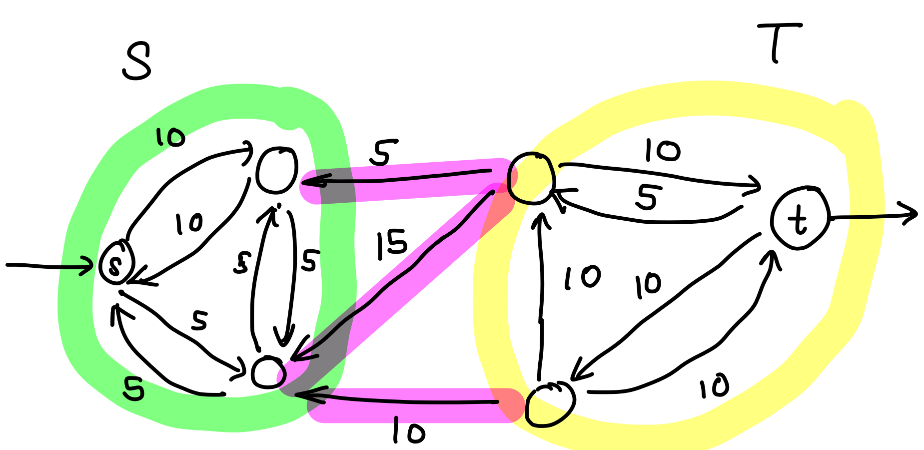
If we push c_{min} units of flow along that path we get a new flow f' s.t $|f'| = |f| + c_{min}$ [Why?]

& f' is feasible [Why?]

$\Longrightarrow$ f is not a max flow!

Case 2 There is no augmenting path from s to t in the residual graph G_f





No augmenting path from s to t!

Let S = all vertices reachable from s in G_f

$T = V \setminus S$

For every vertex $u \in S$, $v \in T$

if $u \to v \in E$, then $f(u \to v) = c(u \to v)$

if $v \to u \in E$, then $f(u \to v) = 0$

Recall, what we saw earlier

If $|f| = \|(S,T)\|$, then

- f is a max-flow
- $S \cup T$ is a min-cut

- $f(u \to v) = c(u \to v)$ for all $u \in S$, $v \in T$
- $f(u \to v) = 0$ for all $u \in T$, $v \in S$

Thus, f is a max-flow & (S,T) is a min-cut

This algorithm was discovered by Ford-Fulkerson in 1952

Augmenting Paths Algorithm

Initialize $f \leftarrow 0$
$G_f \leftarrow G$

While there is a path p from s to t in G_f
push flow along p
rebuild G_f

Return f

Runtime of this algorithm ? NEXT LECTURE