SECTION 6.3

Max-flow min-cut

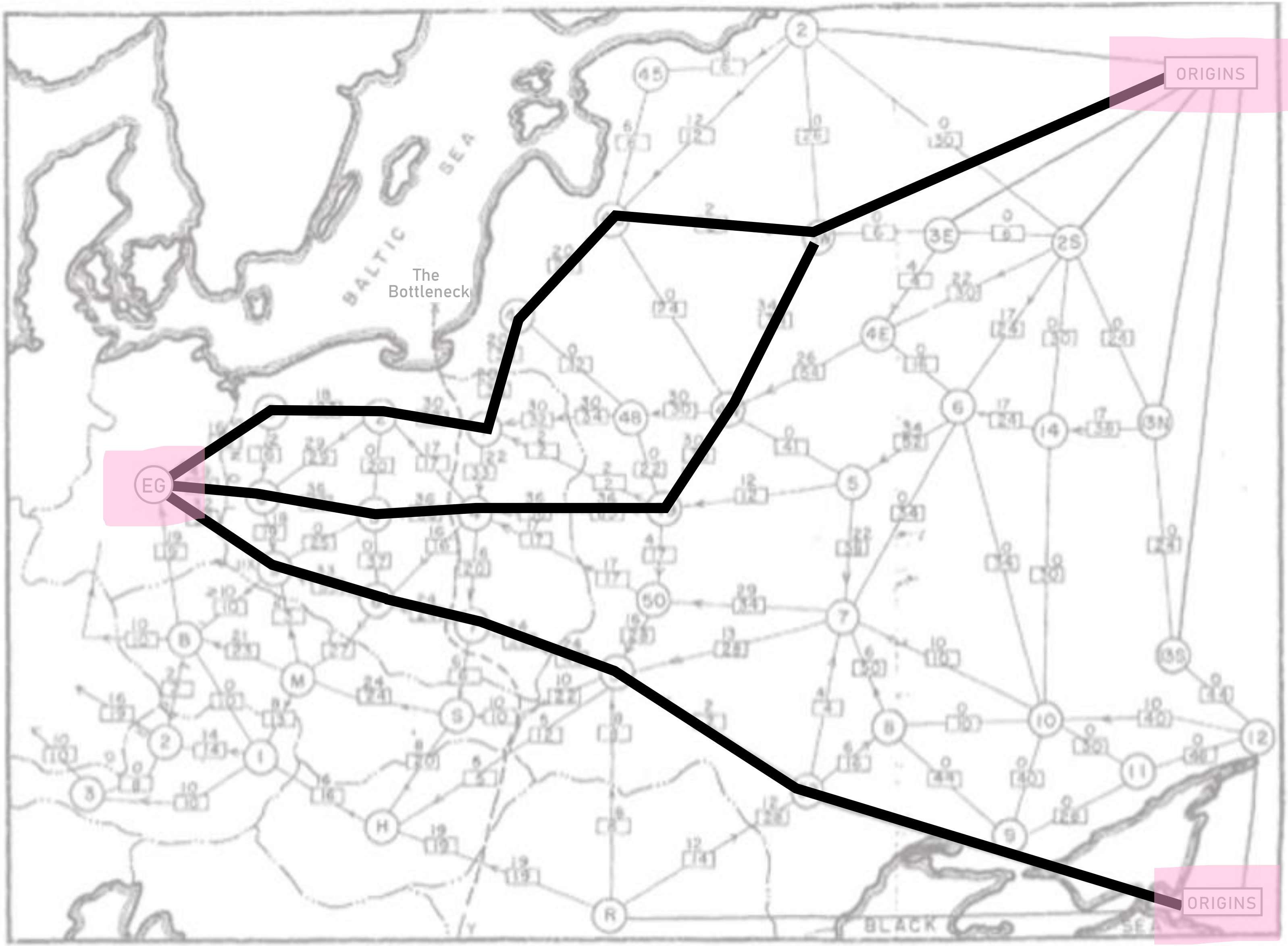


Fig. 7 — Traffic pattern: entire network available

Legend:

International boundary

Railway operating division

Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

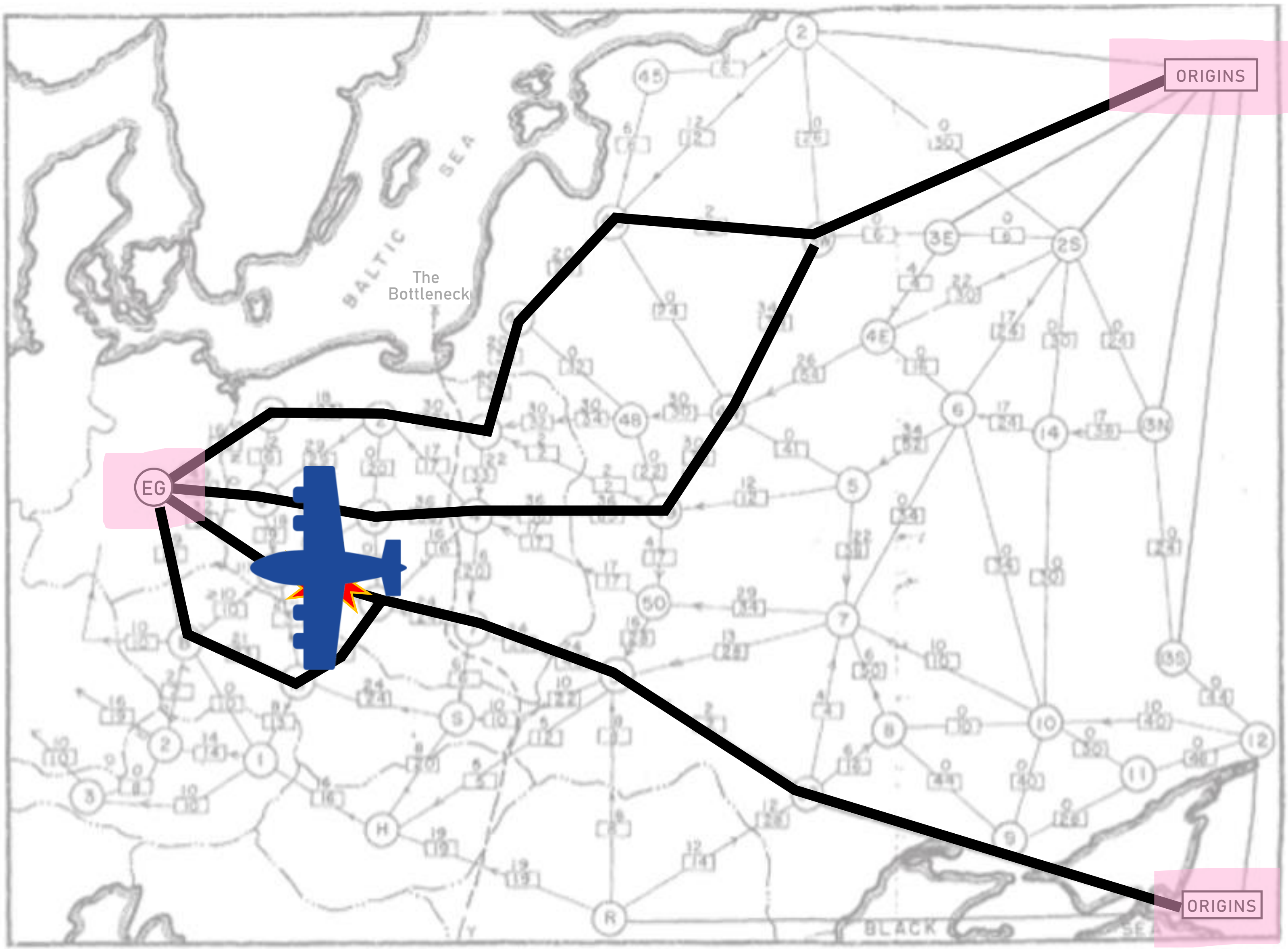
All capacities in trains / $\sqrt{}$1000's of tons } each way per day

Origins: Divisions 2, 3W, 3C, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIK of Division 9, Poland



SECRET RM-1573 10-24-55 -33-

Fig. 7 — Traffic pattern: entire network available

Legend:

International boundary

Railway operating division

Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in trains / 1000's of tons each way per day

Origins: Divisions 2, 3W, 3C, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIK of Division 9, Poland

RM-1573
10-24-55
-33-

SECRET

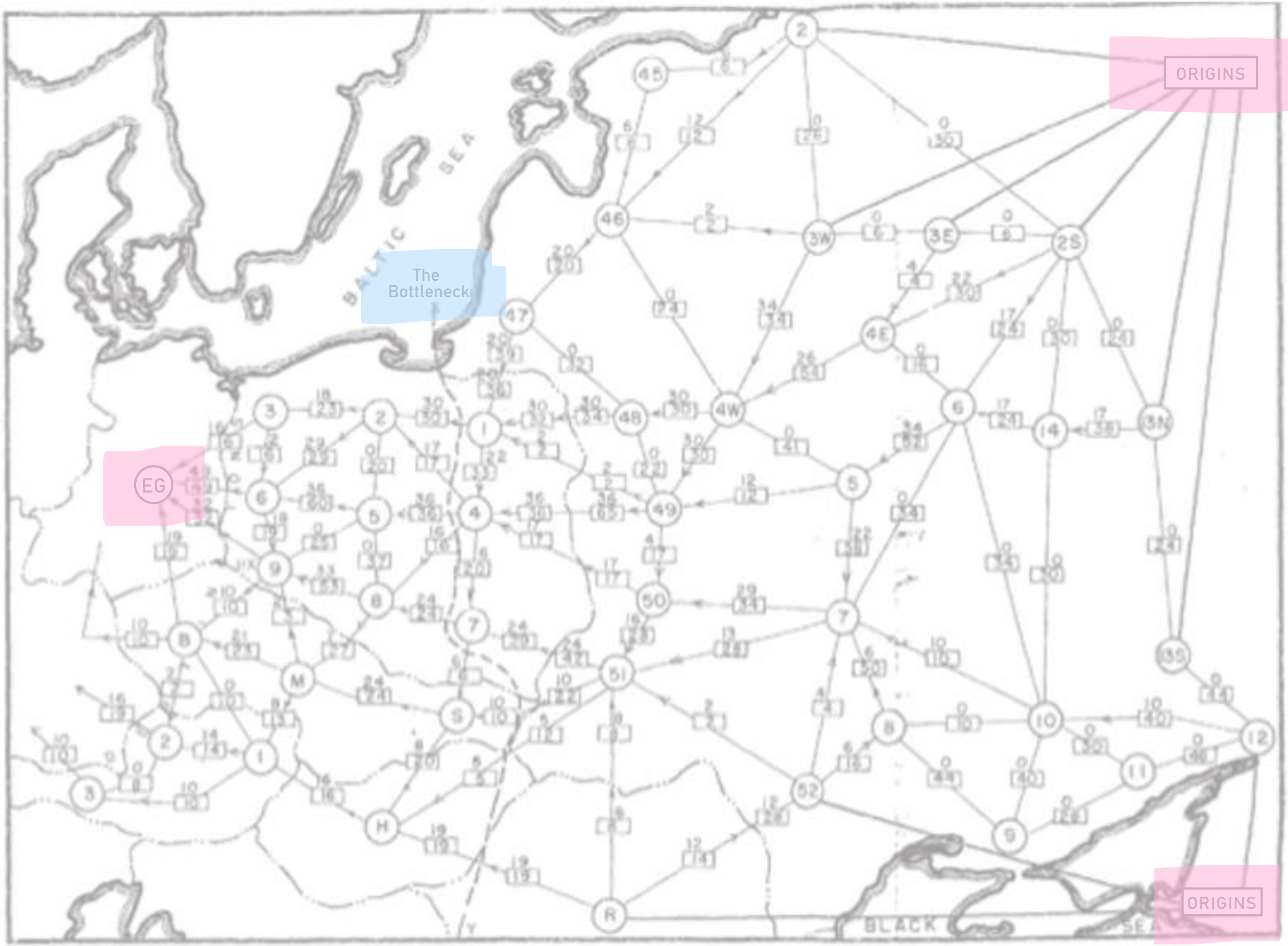


Fig. 7 — Traffic pattern: entire network available

Legend:

International boundary

Railway operating division

Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

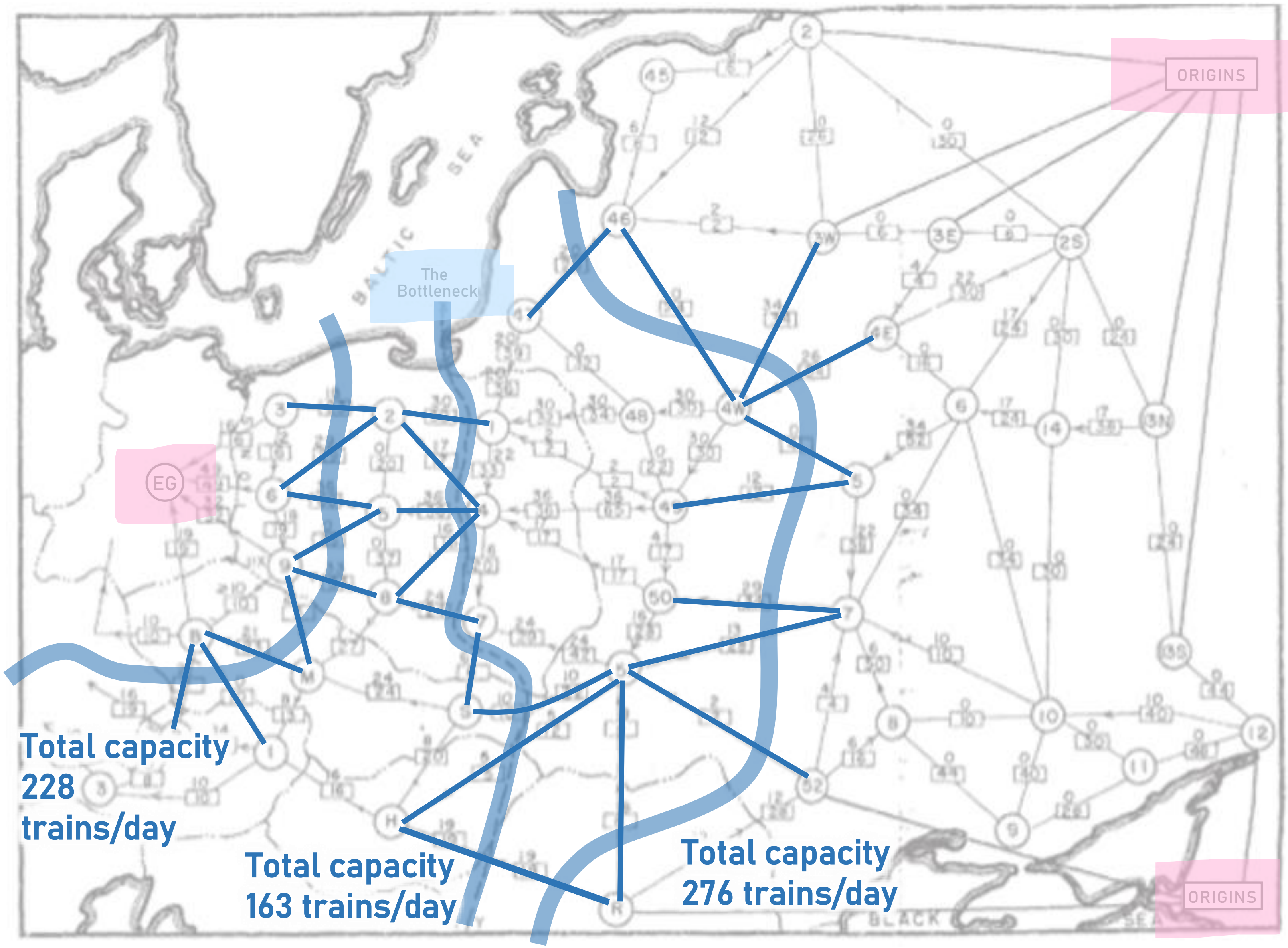
All capacities in trains / 1000's of tons each way per day

Origins: Divisions 2, 3W, 3E, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note 11X of Division 9, Poland



SECRET RM-1573 10-24-55 -33-

Fig. 7 — Traffic pattern: entire network available

Legend:

— · · — International boundary

(8) Railway operating division

9 [12] Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in trains / 1000's of tons each way per day

Origins: Divisions 2, 3W, 3E, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

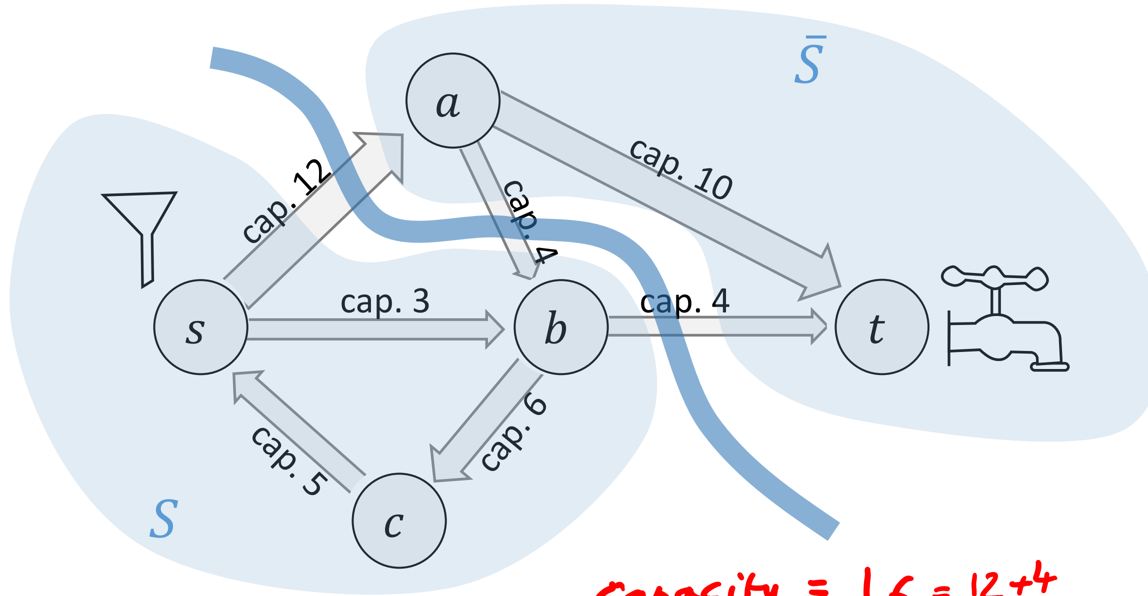
Alternative destinations: Germany or East Germany

Note IIK of Division 9, Poland

A **cut** is a partition of the vertices into two sets, $V = S \cup \bar{S}$, with the source vertex $s \in S$ and the sink vertex $t \in \bar{S}$.

The **capacity** of the cut is

$$\text{capacity}(S, \bar{S}) = \sum_{\substack{u \in S,\, v \in \bar{S}: \\ u \to v}} c(u \to v)$$

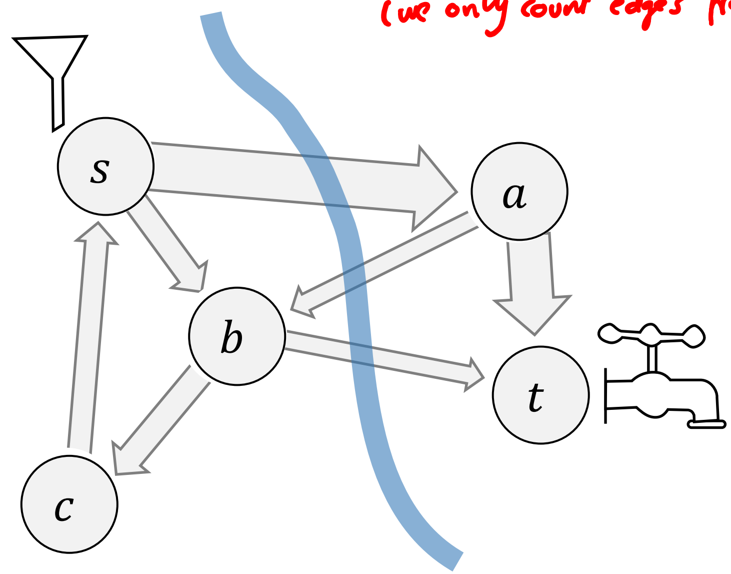


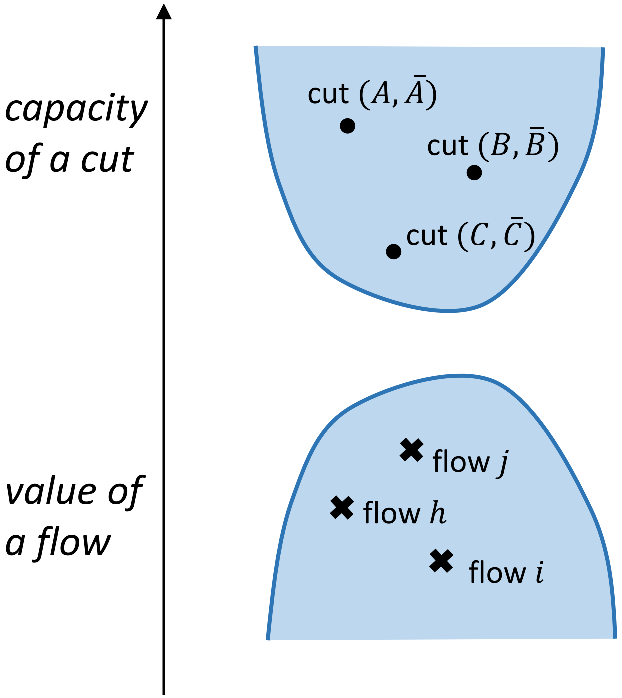
capacity = 16 = 12+4
(we only count edges from S to $\bar{S}$.)

MAX-FLOW MIN-CUT THEOREM

For any flow f and any cut $(S, \bar{S})$,

$$\text{value}(f) \leq \text{capacity}(S, \bar{S})$$

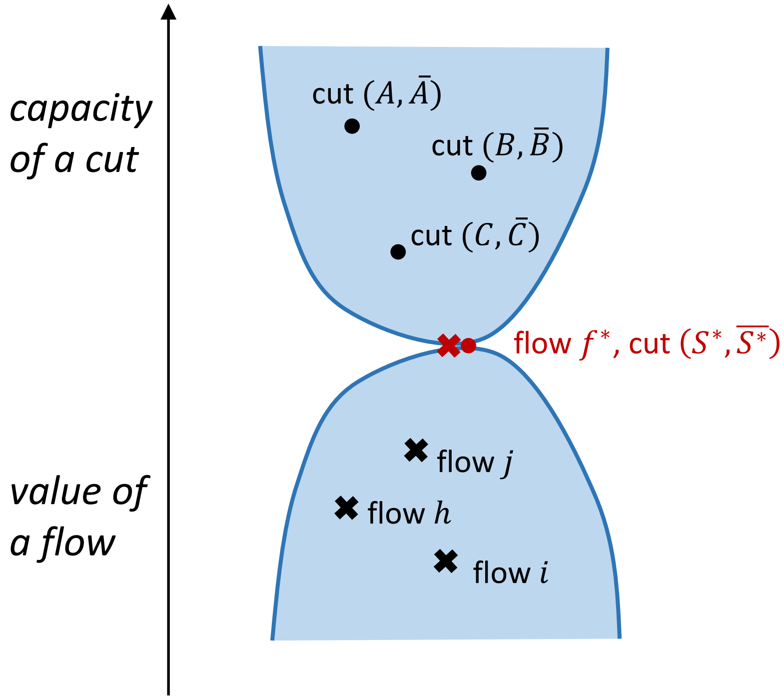




MAX-FLOW MIN-CUT THEOREM

For any flow f and any cut $(S, \bar{S})$,

$$\text{value}(f) \leq \text{capacity}(S, \bar{S})$$



If we can find a flow f^* and a cut $(S^*, \overline{S^*})$ such that value(f^*) = capacity$(S^*, \overline{S^*})$ then f^* is a maximum flow.

Proof let g be any other flow.

value(g) $\leq$ capacity$(S^*, \overline{S^*})$ by MFMC

= value(f^*) by assumption.

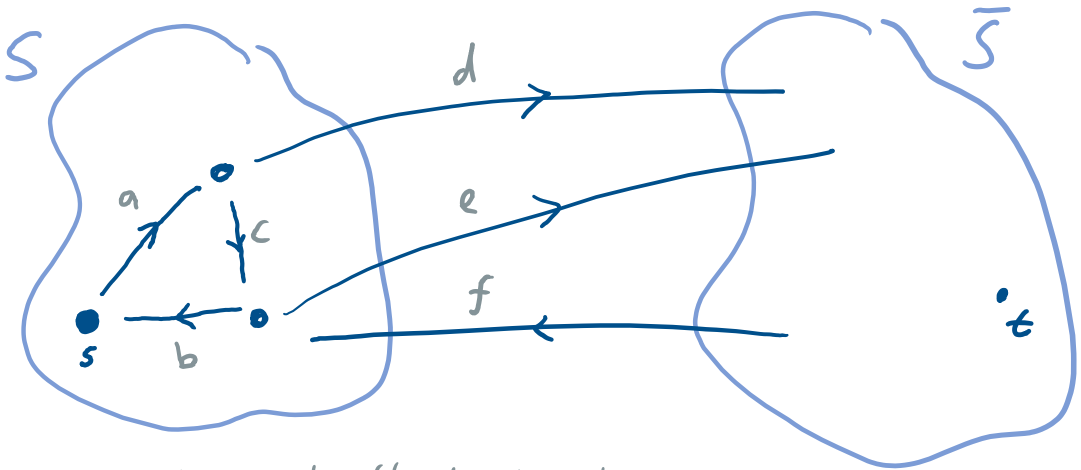
$\therefore$ f^* is a maximum flow

FORD-FULKERSON CLAIM

The Ford-Fulkerson algorithm, if it terminates, finds a flow f^* and a cut $(S^*, \overline{S^*})$ such that

$$\text{value}(f^*) = \text{capacity}(S^*, \overline{S^*})$$

An illustration of the Max-Flow Min-Cut Theorem



Let a, b, etc. be flow amounts.

PROOF STRATEGY
Write out the flow conservation equations for each vertex in $S \setminus \{s\}$, and sum them. Then use

$$0 \leq \text{flow} \leq \text{capacity}$$

Flow conservation at vertices other than s, t:

$$\left.\begin{aligned} a &= d+c \\ c+f &= b+e \end{aligned}\right\} \Rightarrow \left.\begin{aligned} a - c - d &= 0 \\ -b + c - e + f &= 0 \end{aligned}\right\}$$

$$\Rightarrow \quad a - b = d + e - f$$

$$\Rightarrow \quad \underbrace{a - b}_{\text{net flow out of } S} \leq d + e \leq \text{capacity of cut}$$

MAX-FLOW MIN-CUT THEOREM. For any flow f and any cut $(S,\bar{S})$, $\text{value}(f) \leq \text{capacity}(S,\bar{S})$

$$\text{value}(f) = \sum_{u: s\to u} f(s\to u) - \sum_{u: u\to s} f(u\to s)$$

by definition of flow value = net flow out = tot flow out − tot flow in

$$= \sum_{u\in V} f(s\to u) - \sum_{u\in V} f(u\to s)$$

where we've extended f to all pairs of vertices, and set $f(v\to w)=0$ if there's no edge $v\to w$

$$= \sum_{v\in S} \left[\sum_{u\in V} f(v\to u) - \sum_{u\in V} f(u\to v) \right]$$

Flow conservation says that the term $[\cdot]$ is 0 for all vertices in $V\setminus\{s,t\}$. We're summing over $v\in S$, so we pick up the $v=s$ contribution, but not t.

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

expanding $\sum_{u\in V} = \sum_{u\in S} + \sum_{u\notin S}$

$$= \cancel{\sum_{a\in S}\sum_{b\in S} f(a\to b)} + \sum_{v\in S}\sum_{u\notin S} f(v\to u) - \cancel{\sum_{b\in S}\sum_{a\in S} f(a\to b)} - \sum_{v\in S}\sum_{u\notin S} f(u\to v)$$

relabelling the "loop" variables

These two terms are identical, so they cancel out. [Similar to the "telescopic sum" in Johnson's alg.]

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

$$\leq \sum_{v\in S}\sum_{u\notin S} f(v\to u)$$

since $f\geq 0$ on every edge

$$\leq \sum_{v\in S}\sum_{u\notin S} c(v\to u)$$

since $f\leq c$ on every edge

$$= \text{capacity}(S,\bar{S})$$

by definition of cut capacity

Important bits of the proof:

$$\text{value}(f) \le \sum_{v \in S} \sum_{u \notin S} f(v \to u) \quad \text{since } f \ge 0 \text{ on every edge}$$

$$\le \sum_{v \in S} \sum_{u \notin S} c(v \to u) \quad \text{since } f \le c \text{ on every edge}$$

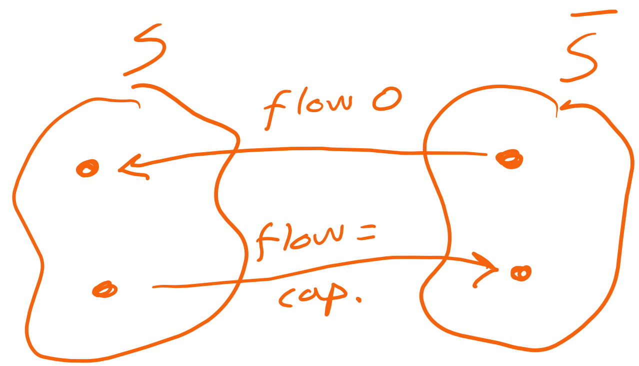
$$= \text{capacity}(S, \bar{S}) \quad \text{by definition of cut capacity}$$

REMARK.

If $f = 0$ on every edge from $\bar{S}$ to S then the first inequality is an equality.

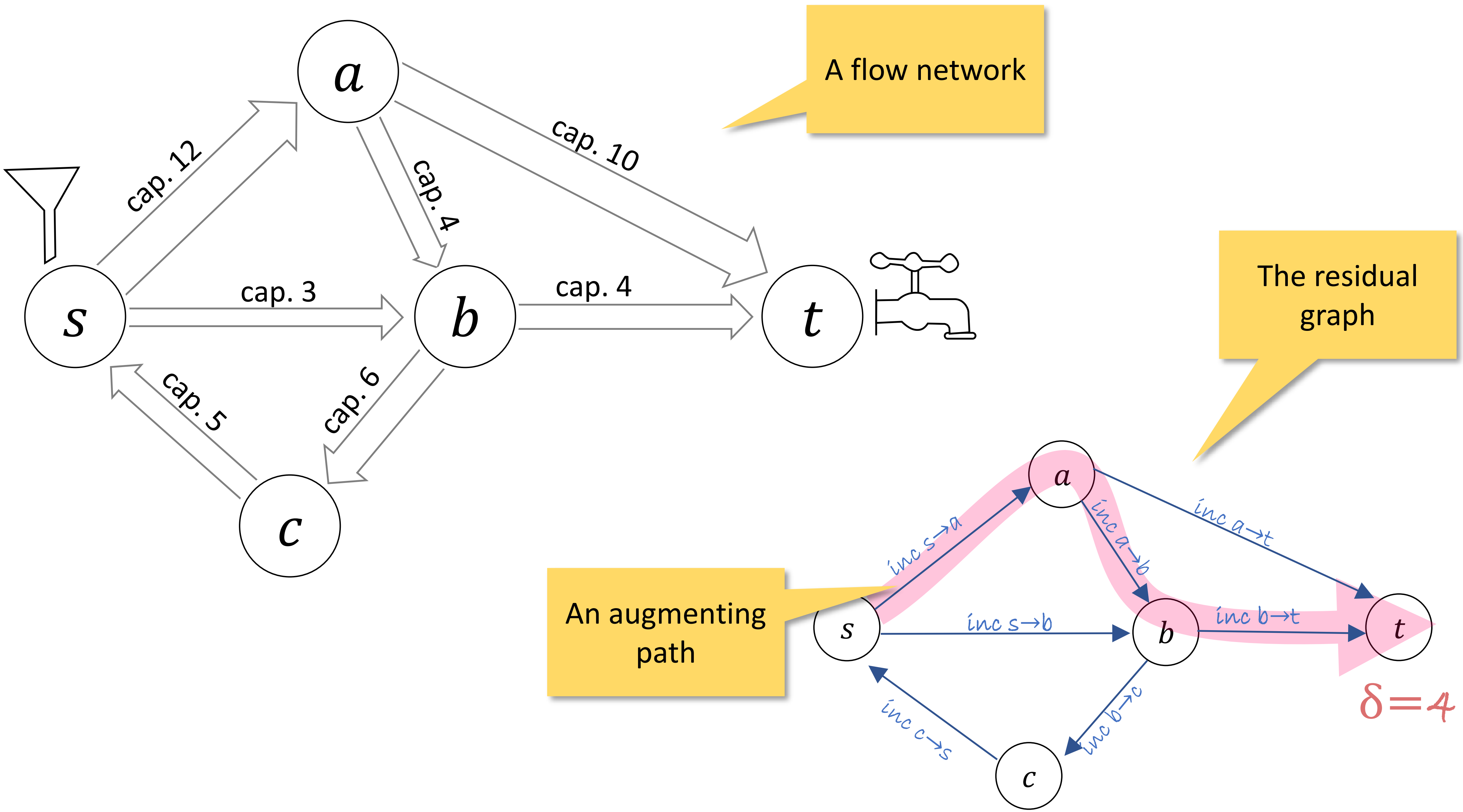
If $f = c$ on every edge from S to $\bar{S}$ then the second inequality is an equality

And if both conditions are met, then $\text{value}(f) = \text{capacity}(S, \bar{S})$.

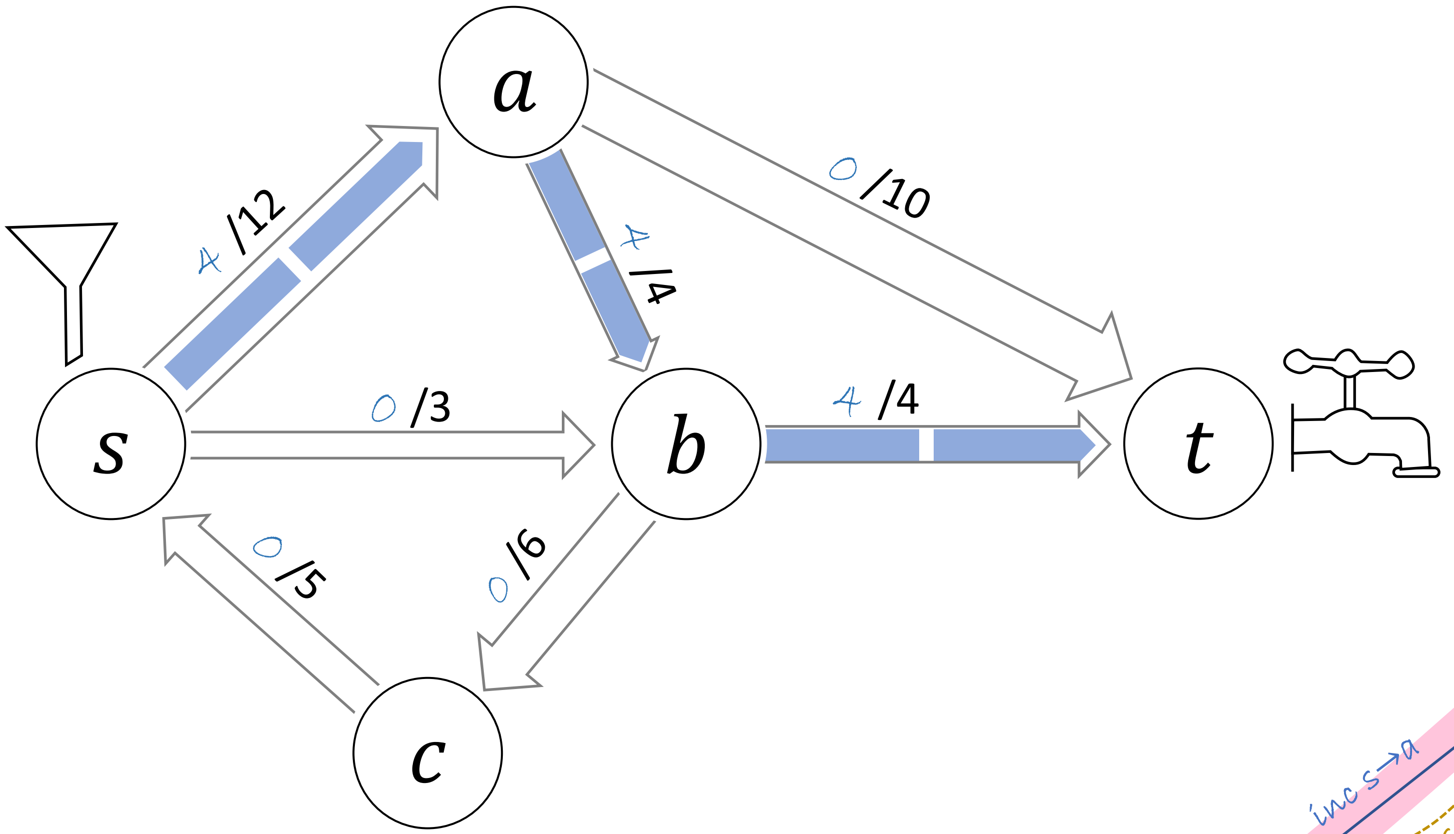


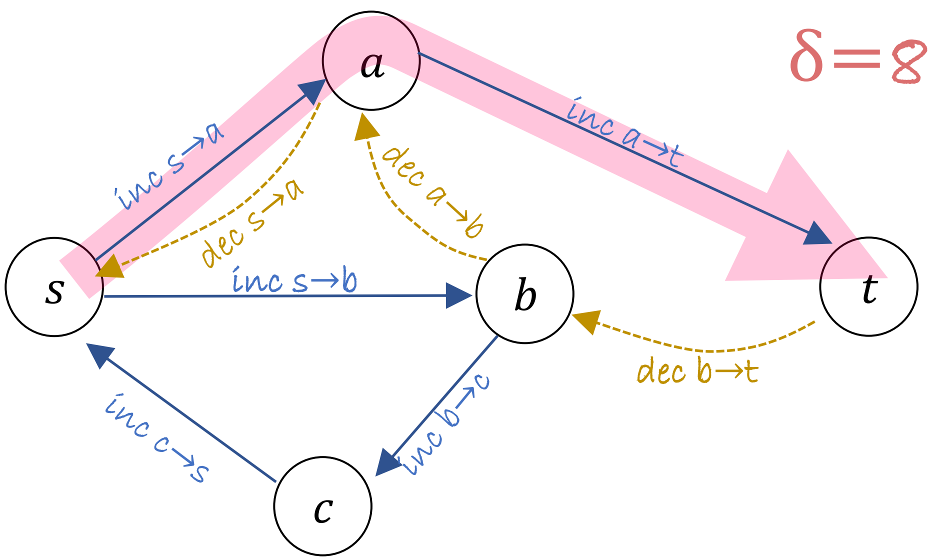
This is how we'll prove the Ford-Fulkerson claim: We'll demonstrate a flow f^* and a cut $(S^*, \overline{S^*})$ such that all edges $\overline{S^*} \to S^*$ have zero flow, and all edges $S^* \to \overline{S^*}$ are at capacity.

WALKTHROUGH OF FORD-FULKERSON

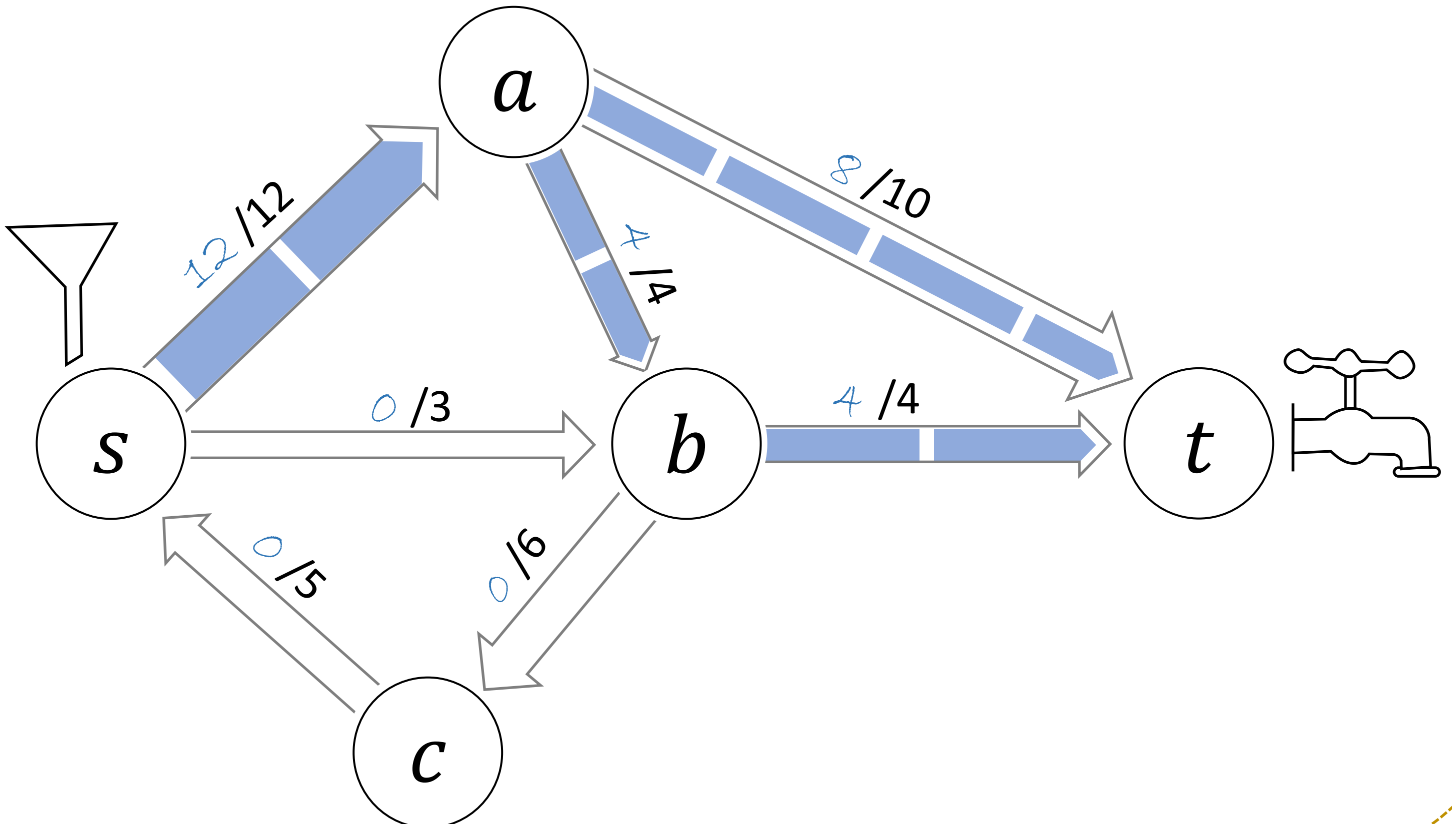


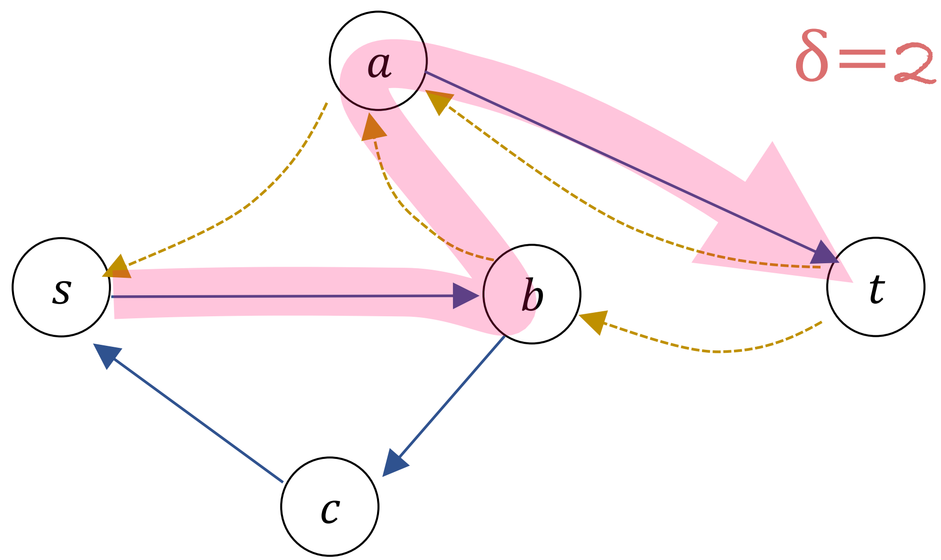
WALKTHROUGH OF FORD-FULKERSON





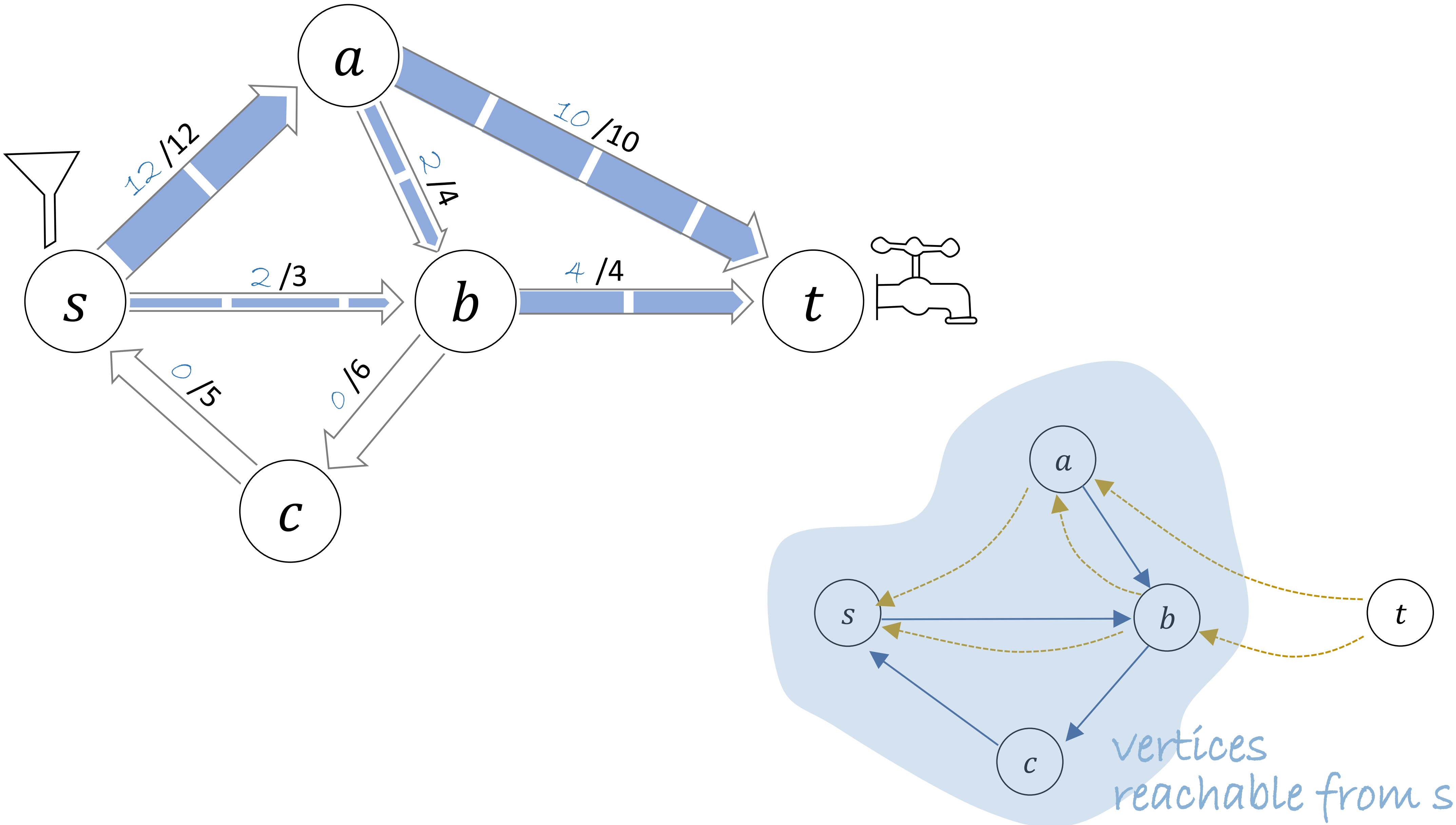
WALKTHROUGH OF FORD-FULKERSON





WALKTHROUGH OF FORD-FULKERSON

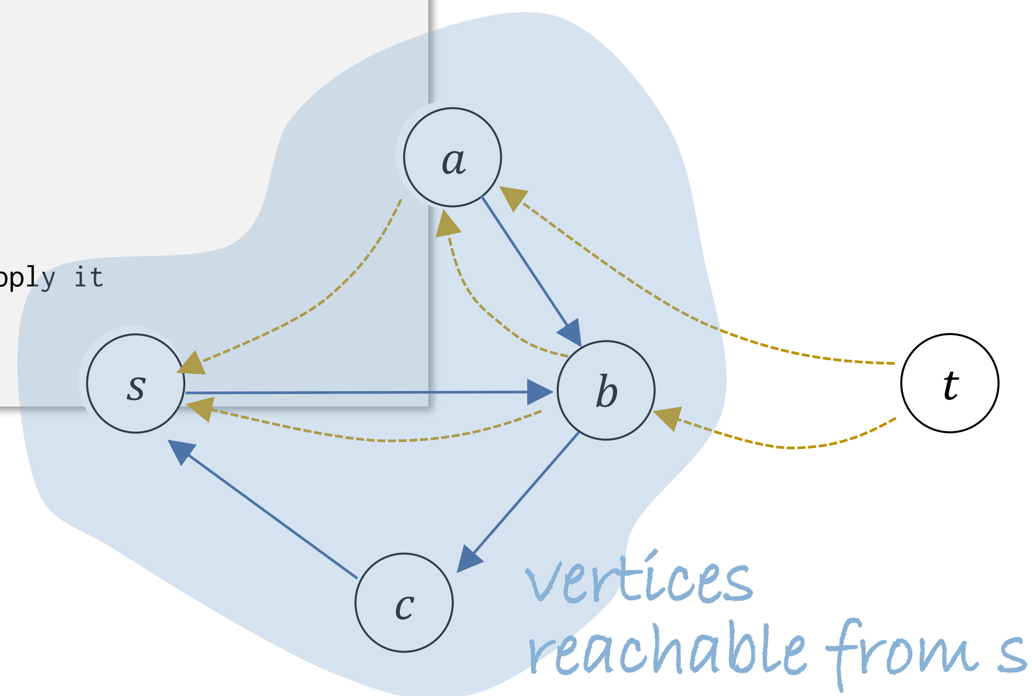
We cannot find an augmenting path in the residual graph. So, terminate.



```
def ford_fulkerson(g, s, t):
    # Let f be a flow, initially empty
    for u → v in g.edges:
        f(u → v) = 0

    # Define a helper function for finding an augmenting path
    def find_augmenting_path():
        # Define the residual graph h on the same vertices as g
        for u → v in g.edges:
            if f(u → v) < c(u → v): give h an edge u → v labelled “inc u → v”
            if f(u → v) > 0: give h an edge v → u labelled “dec u → v”
        if h has a path from s to t:
            return some such path, together with the labels of its edges
        else:
            # Let S be the set of vertices reachable from s (used in the proof)
            return None

    # Repeatedly find an augmenting path and add flow to it
    while True:
        p = find_augmenting_path()
        if p is None:
            break
        else:
            compute δ, the amount of flow to apply along p, and apply it
            # Assert: δ > 0
            # Assert: f is still a valid flow
```



FORD-FULKERSON CLAIM

The Ford-Fulkerson algorithm, if it terminates, finds a flow f^* and a cut $(S^*, \overline{S^*})$ such that value(f^*) = capacity$(S^*, \overline{S^*})$

Proof

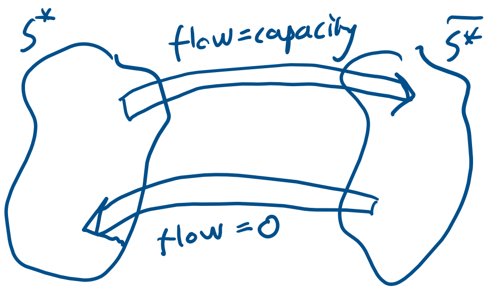
Suppose it terminates. Let f^* be the final flow it produces, and

let $S^* = \{$ vertices reachable from s in the residual graph, at termination $\}$.

Then $(S^*, \overline{S^*})$ is a cut.

(Recall the definition of a cut: we need $s \in S^*$ and $t \in \overline{S^*}$. This is so because, at termination, we can't reach t.)

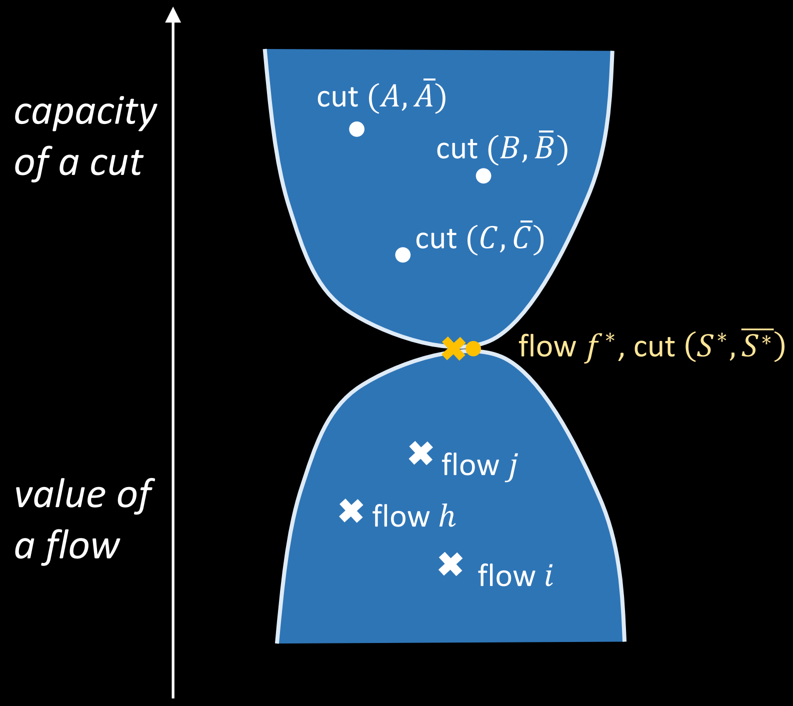
And the residual graph has no edges $S^* \to \overline{S^*}$ (because vertices in $\overline{S^*}$ are unreachable, by defn. of S^*)



- If the capacity graph has an edge $v^{\in S^*} \to u^{\overline{S^*}}$ then $f^*(v \to u) = c(v \to u)$
  (otherwise the resid. graph would have an edge $v \xrightarrow{\text{inc. } v \to u} u$)
- If the capacity graph has an edge $u^{\in \overline{S^*}} \to v^{\in S^*}$ then $f^*(u \to v) = 0$
  (otherwise the resid. graph would have an edge $v \xrightarrow{\text{dec. } u \to v} u$)

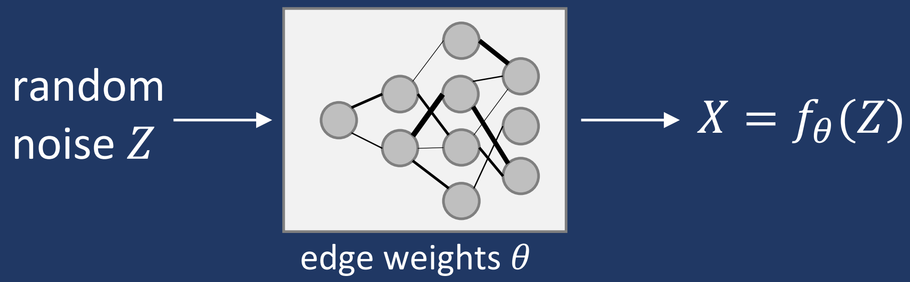
By the remark at the end of the proof of Max-Flow Min-Cut Theorem,

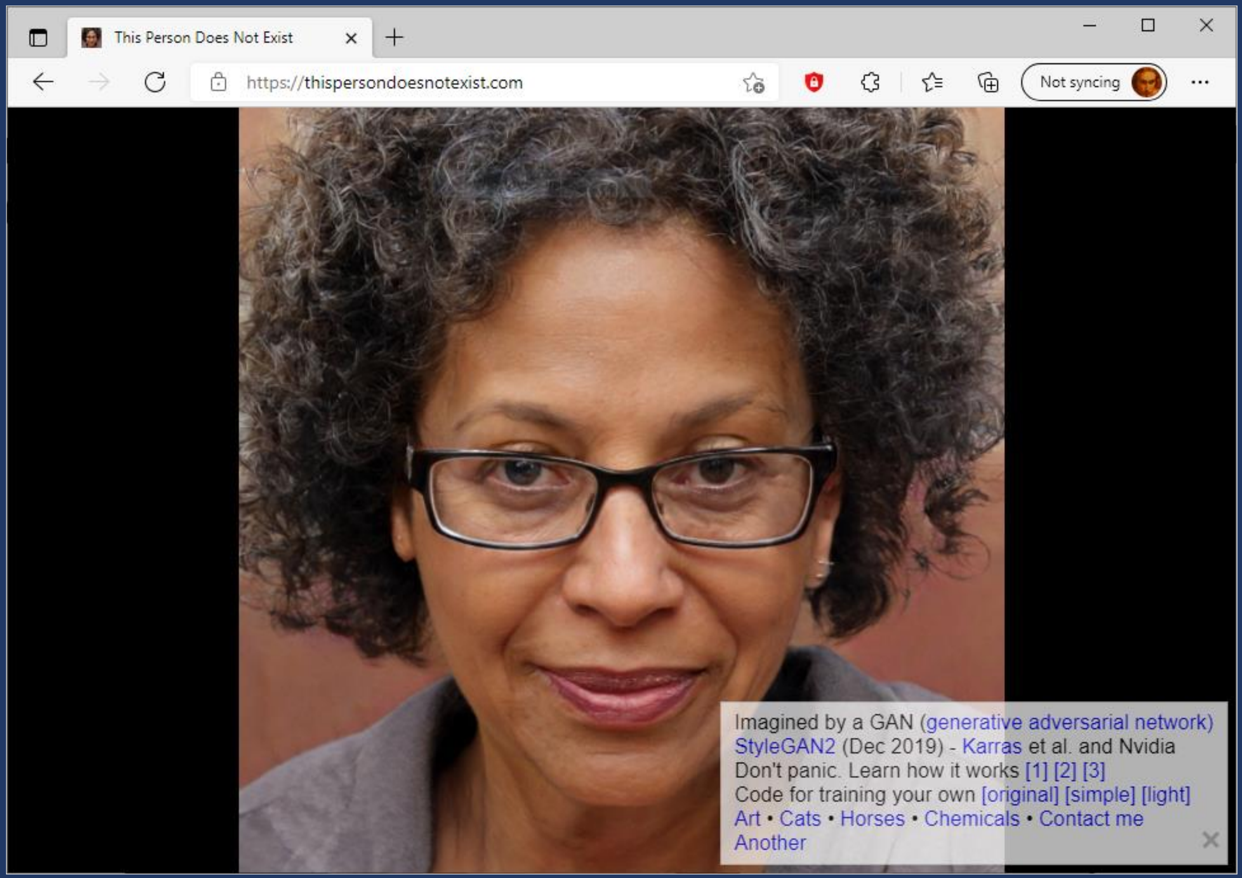
value(f^*) = capacity$(S^*, \overline{S^*})$.



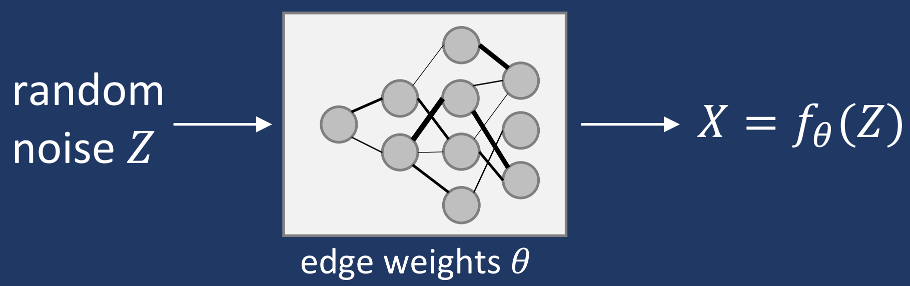
- The Ford-Fulkerson algorithm produces both a flow and a cut; and the cut acts as a *certificate of optimality* for the flow.
- Many other optimization algorithms also produce a (solution, certificate) pair. The certificate corresponds to the dual variables in Lagrangian optimization.

A *latent generative model* is a neural network that has been trained to map a random noise vector into something that resembles items from the training dataset.

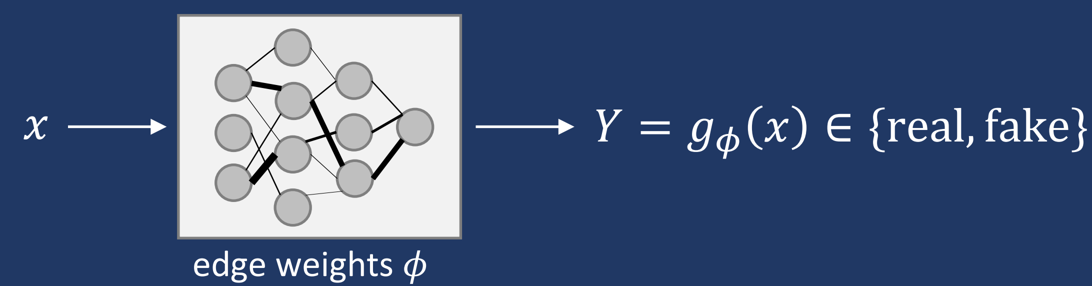




A *latent generative model* is a neural network that has been trained to map a random noise vector into something that resembles items from the training dataset.



An *adversary* is a neural network that guesses whether an input x is real (i.e. from the training dataset) or fake (i.e. generated by us).



We can train a good generator by simultaneously training an adversary. When we've finished training, the adversary should be unable to detect whether a given x is real or fake. The adversary is a *certificate* that our generator is good.

Department of Computer Scien

cl.cam.ac.uk/teaching/2324/Algorithm1/materials.html

5. Graphs and path finding

Lecture 09 [slides]	5, 5.1 Graphs (14:27) 5.2 Depth-first search (11:37) 5.3 Breadth-first search (6:43)
Lecture 10 [slides]	5.4 Dikstra's algorithm (15:25) pl proof (24:01)
Lecture 11 [slides.pre]	5.5 Algorithms and proofs (9:29) 5.6 Bellman-Ford (12:13)
Lecture 12 [slides.pre]	5.7 Dynamic programming (13:06 5.8 Johnson's algorithm (13:43)

Example sheet 4 [pdf]
Optional tick: bfs-all from ex4.q6
Optional challenge: chatgpt-bfs
Optional assignment: grade-chatgpt
Optional tick: bf-cycle from ex4.q19

Algorithms task grade-chatgpt

cl.cam.ac.uk/teaching/2324/Algorithm1/cl/chatgpt/grade-chatgpt.html

Algorithms assignment grade-gpt: Grading ChatGPT's proof

Can ChatGPT be persuaded to give a proper proof of correctness of an algorithm? Here are three attempts, for an algorithm that solves the bfs-all tick:

- Catley, Prynn, and Huang.

Please mark these attempts, on a scale of 0–20. Your mark should be for the final proof, not for how well it was elicited. **Please submit your grades on Moodle.** I'll pick the most controversially-marked answer and go through it in lectures. Please use the following marking scheme:

mark	meaning
5	Coherent fragments
9	Coherent in parts, but with serious gaps
13	A basically correct argument but with some signs of confusion
17	Essentially correct, but not fully rigorous
19	Nearly all correct, only minor technical holes

How well does ChatGPT generate algorithms? For interest, here are the attempts to get ChatGPT to design the algorithm:

- Catley, Shen, Chen, Prynne, and Huang.