Aside: Matrix Multiplication

Admin

- Midterm grades ready to be viewed
  - Went really well from my perspective
- Assignment 6 out tonight
- CS Grad school colloquium today at 3:15
- Anything else?

Matrix Multiplication

Problem. Given two n-by-n matrices A and B, compute matrix $C = A \cdot B$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

Standard multiplication computes each c_{ij} as:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Complexity. $\Theta(n^3)$ operations (scalar multiplications)

Block Matrix Multiplication

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$$

C_{11} A_{11} A_{12} B_{11}

$$\begin{bmatrix} 152 & 158 & 164 & 170 \\ 504 & 526 & 548 & 570 \\ 856 & 894 & 932 & 970 \\ 1208 & 1262 & 1316 & 1370 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \times \begin{bmatrix} 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 \\ 28 & 29 & 30 & 31 \end{bmatrix}$$

B_{21}

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} = \begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \times \begin{bmatrix} 16 & 17 \\ 20 & 21 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \times \begin{bmatrix} 24 & 25 \\ 28 & 29 \end{bmatrix} = \begin{bmatrix} 152 & 158 \\ 504 & 526 \end{bmatrix}$$

Block Matrix Multiplication

To multiply two n-by-n matrices A and B:

- **Divide**: partition A and B into $\frac{n}{2}$ by $\frac{n}{2}$ matrices
- **Conquer**: multiply 8 pairs of $\frac{n}{2}$ by $\frac{n}{2}$ matrices recursively
- **Combine**: Add products using 4 matrix additions

n-by-n matrices

$$C = A \times B$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

½n-by-½n matrices

8 matrix multiplications
(of ½n-by-½n matrices)

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

4 matrix additions
(of ½n-by-½n matrices)

Block Matrix Multiplication

Running time recurrence.

- $T(n) = 8T(n/2) + \Theta(n^2)$
- How do we solve it with the recursion-tree method?
  - $T(n) = O(n^3)$

- Nice idea but it didn't improve the run time, oh well!
- Divide and conquer version is still more **cache-efficient**

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

½n-by-½n matrices

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

4 matrix additions
(of ½n-by-½n matrices)

Block MM: Strassen's Trick

Key idea. Can multiply two 2-by-2 matrices via 7 scalar multiplications (plus 11 additions and 7 subtractions).

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned}
P_1 &\leftarrow A_{11} \times (B_{12} - B_{22}) \\
P_2 &\leftarrow (A_{11} + A_{12}) \times B_{22} \\
P_3 &\leftarrow (A_{21} + A_{22}) \times B_{11} \\
P_4 &\leftarrow A_{22} \times (B_{21} - B_{11}) \\
P_5 &\leftarrow (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\
P_6 &\leftarrow (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\
P_7 &\leftarrow (A_{11} - A_{21}) \times (B_{11} + B_{12})
\end{aligned}$$

7 scalar multiplications

$$\begin{aligned}
C_{11} &= P_5 + P_4 - P_2 + P_6 \\
C_{12} &= P_1 + P_2 \\
C_{21} &= P_3 + P_4 \\
C_{22} &= P_1 + P_5 - P_3 - P_7
\end{aligned}$$

Pf.
$$\begin{aligned}
C_{12} &= P_1 + P_2 \\
&= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22} \\
&= A_{11} \times B_{12} + A_{12} \times B_{22}.
\end{aligned}$$

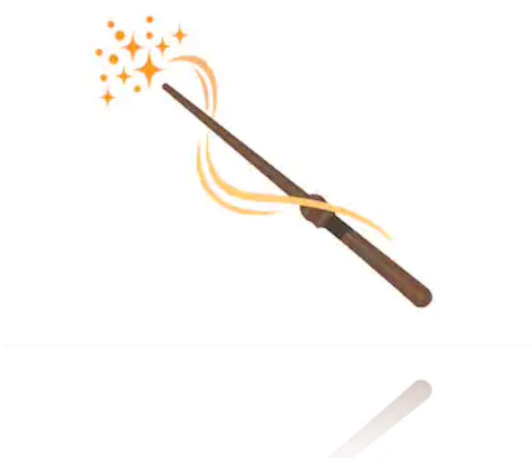
Block MM: Strassen's Trick

Key idea. Can multiply two n-by-n matrices via 7 $n/2$-by-$n/2$ matrix multiplications (plus 11 additions and 7 subtractions).

½n-by-½n matrices

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_1 + P_5 - P_3 - P_7 \end{aligned}$$



$$\begin{aligned} P_1 &\leftarrow A_{11} \times (B_{12} - B_{22}) \\ P_2 &\leftarrow (A_{11} + A_{12}) \times B_{22} \\ P_3 &\leftarrow (A_{21} + A_{22}) \times B_{11} \\ P_4 &\leftarrow A_{22} \times (B_{21} - B_{11}) \\ P_5 &\leftarrow (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\ P_6 &\leftarrow (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\ P_7 &\leftarrow (A_{11} - A_{21}) \times (B_{11} + B_{12}) \end{aligned}$$

7 recursive multiplications

Pf. $$\begin{aligned} C_{12} &= P_1 + P_2 \\ &= A_{11} \times (B_{12} - B_{22}) + (A_{11} + A_{12}) \times B_{22} \\ &= A_{11} \times B_{12} + A_{12} \times B_{22}. \end{aligned}$$

Strassen's MM Algorithm

assume n is a power of 2

$\textsc{Strassen}(n, A, B)$

$\textsc{If}$ $(n = 1)$ $\textsc{Return}$ $A \times B$.

Partition A and B into ½n-by-½n blocks.

$P_1 \leftarrow \textsc{Strassen}(n / 2, A_{11}, (B_{12} - B_{22}))$.

$P_2 \leftarrow \textsc{Strassen}(n / 2, (A_{11} + A_{12}), B_{22})$.

$P_3 \leftarrow \textsc{Strassen}(n / 2, (A_{21} + A_{22}), B_{11})$.

$P_4 \leftarrow \textsc{Strassen}(n / 2, A_{22}, (B_{21} - B_{11}))$.

$P_5 \leftarrow \textsc{Strassen}(n / 2, (A_{11} + A_{22}), (B_{11} + B_{22}))$.

$P_6 \leftarrow \textsc{Strassen}(n / 2, (A_{12} - A_{22}), (B_{21} + B_{22}))$.

$P_7 \leftarrow \textsc{Strassen}(n / 2, (A_{11} - A_{21}), (B_{11} + B_{12}))$.

$7\,T(n / 2) + \Theta(n^2)$

$C_{11} = P_5 + P_4 - P_2 + P_6$.

$C_{12} = P_1 + P_2$.

$C_{21} = P_3 + P_4$.

$C_{22} = P_1 + P_5 - P_3 - P_7$.

$\Theta(n^2)$

$\textsc{Return}$ C.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Strassen’s MM Algorithm Analysis

- We get the following recurrence
    - $T(n) = 7T(n/2) + \Theta(n^2)$
- What does the running time recurrence solve to?
    - We have a increasing geometric series
    - Thus, the cost is dominated by the leaves
    - $T(n) = \Theta(r^L) = \Theta(7^{\log_2 n}) = \Theta(n^{\log_2 r}) \approx \Theta(n^{2.81})$
    - We have a much “faster” algorithm!

History of Matrix Multiplication

year	algorithm	arithmetic operations
1858	**“grade school”**	$O(n^3)$
1969	**Strassen**	$O(n^{2.808})$
1978	**Pan**	$O(n^{2.796})$
1979	**Bini**	$O(n^{2.780})$
1981	**Schönhage**	$O(n^{2.522})$
1982	**Romani**	$O(n^{2.517})$
1982	**Coppersmith-Winograd**	$O(n^{2.496})$
1986	**Strassen**	$O(n^{2.479})$
1989	**Coppersmith-Winograd**	$O(n^{2.3755})$
2010	**Strother**	$O(n^{2.3737})$
2011	**Williams**	$O(n^{2.372873})$
2014	**Le Gall**	$O(n^{2.372864})$
2021	**Alman-Williams**	$O(n^{2.37286})$

galactic algorithms

“Galactic algorithm: runs faster than any other algorithm for problems that are sufficiently large, but "sufficiently large" is so big that the algorithm is never used in practice.”

How fast can matrix multiplication get?

- Best lower bound: $\Omega(n^2)$
- Known methods cannot get better than ~ $\Omega(n^{2.37})$
- If we allow $O(1)$ time arithmetic on arbitrarily large integers, can get $O(n^2)$ [Han, Unpublished]
- Why do we care?
  - Important for practice if makes things faster
  - New methods for new bounds

Why are we doing this?

Why are we doing this?

- Not because the fastest exponent = fastest method in practice
- New methods that can be useful in other contexts
- Paves the way for other methods that are fast in practice
- Better understanding of what computers can do

Matrix Multiplication in Practice

- Is Strassen's worth it?
- Strassen's is better than a simple MM implementation (use 3 loops to compute all sums)
- But it's (generally) a little worse than a $O(n^3)$ time hyper-optimized MM implementation

Tons of Applications

- Lots of problem reduce to matrix multiplication complexity

linear algebra problem	expression	arithmetic complexity
matrix multiplication	$A \times B$	$MM(n)$
matrix squaring	A^2	$\Theta(MM(n))$
matrix inversion	A^{-1}	$\Theta(MM(n))$
determinant	$\lvert A \rvert$	$\Theta(MM(n))$
rank	$rank(A)$	$\Theta(MM(n))$
system of linear equations	$Ax = b$	$\Theta(MM(n))$
LU decomposition	$A = L\,U$	$\Theta(MM(n))$
least squares	$\min \lVert Ax - b \rVert_2$	$\Theta(MM(n))$

numerical linear algebra problems with the same arithmetic complexity MM(n) as matrix multiplication

And nontrivial applications

- Triangle finding/clique finding in a graph
- “Lightbulb” problem (find correlations between long random vectors)
- String matching

Introduction to Network Flows

New Algorithmic Paradigm

- **Network flows** model a variety of optimization problems
- These optimization problems look complicated with lots of constraints and on the face of it have nothing to do with networks
- Very powerful problem solving frameworks
- We'll focus on the concept of **problem reductions**
    - Problem A reduces to B if a solution to B leads to a solution to A
- Learn how to prove that our reductions are correct

Network Flow History

- In 1950s, US military researchers Harris and Ross wrote a classified report about the rail network linking Soviet Union and Eastern Europe
    - Vertices were the geographic regions
    - Edges were railway links between the regions
    - Edge weights were the rate at which material could be shipped from one region to next
- Ross and Harris determined:
    - Maximum amount of stuff that could be moved from Russia to Europe **(max flow)**
    - Cheapest way to disrupt the network by removing rail links **(min cut)**

Network Flow History

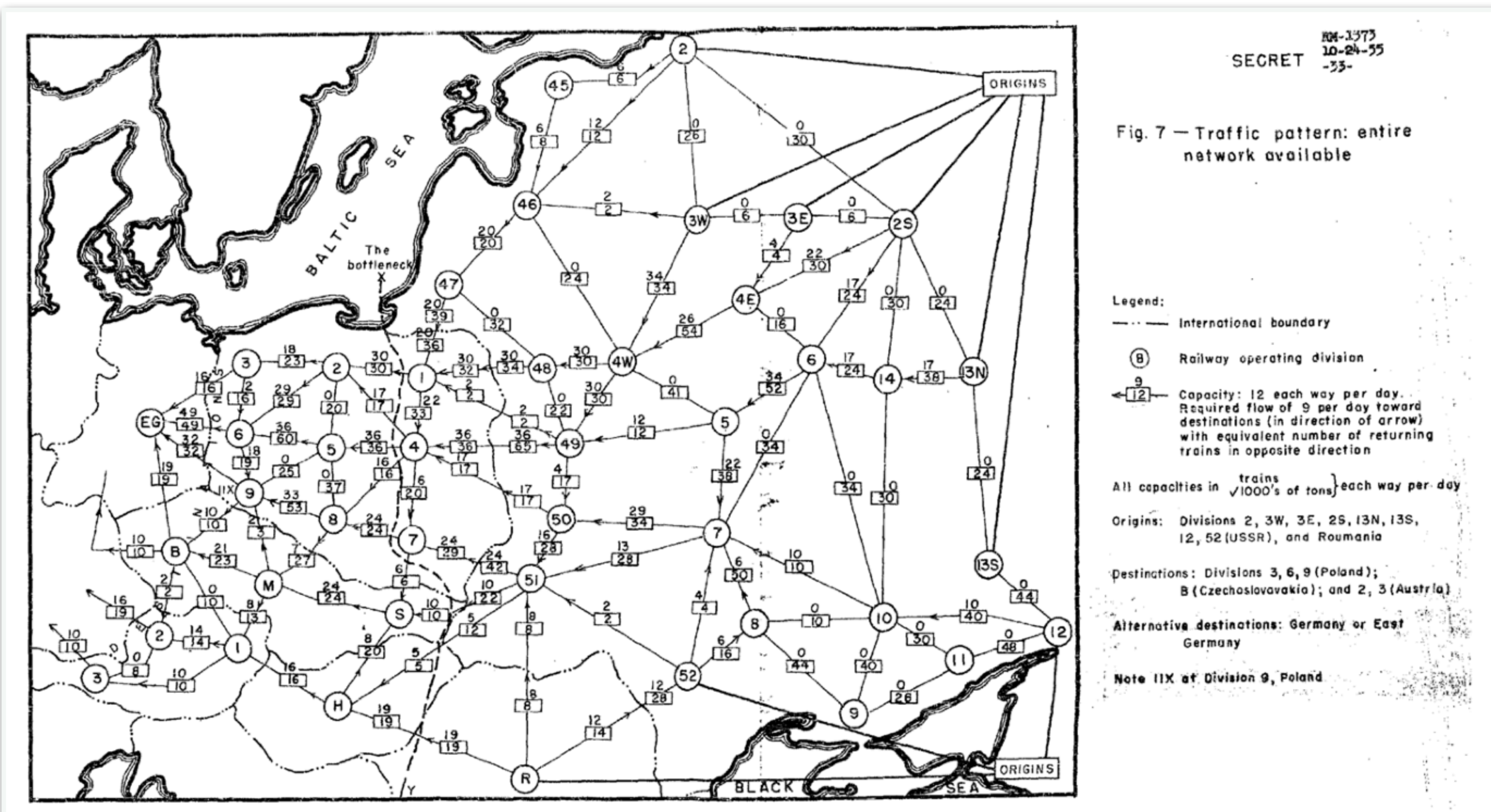
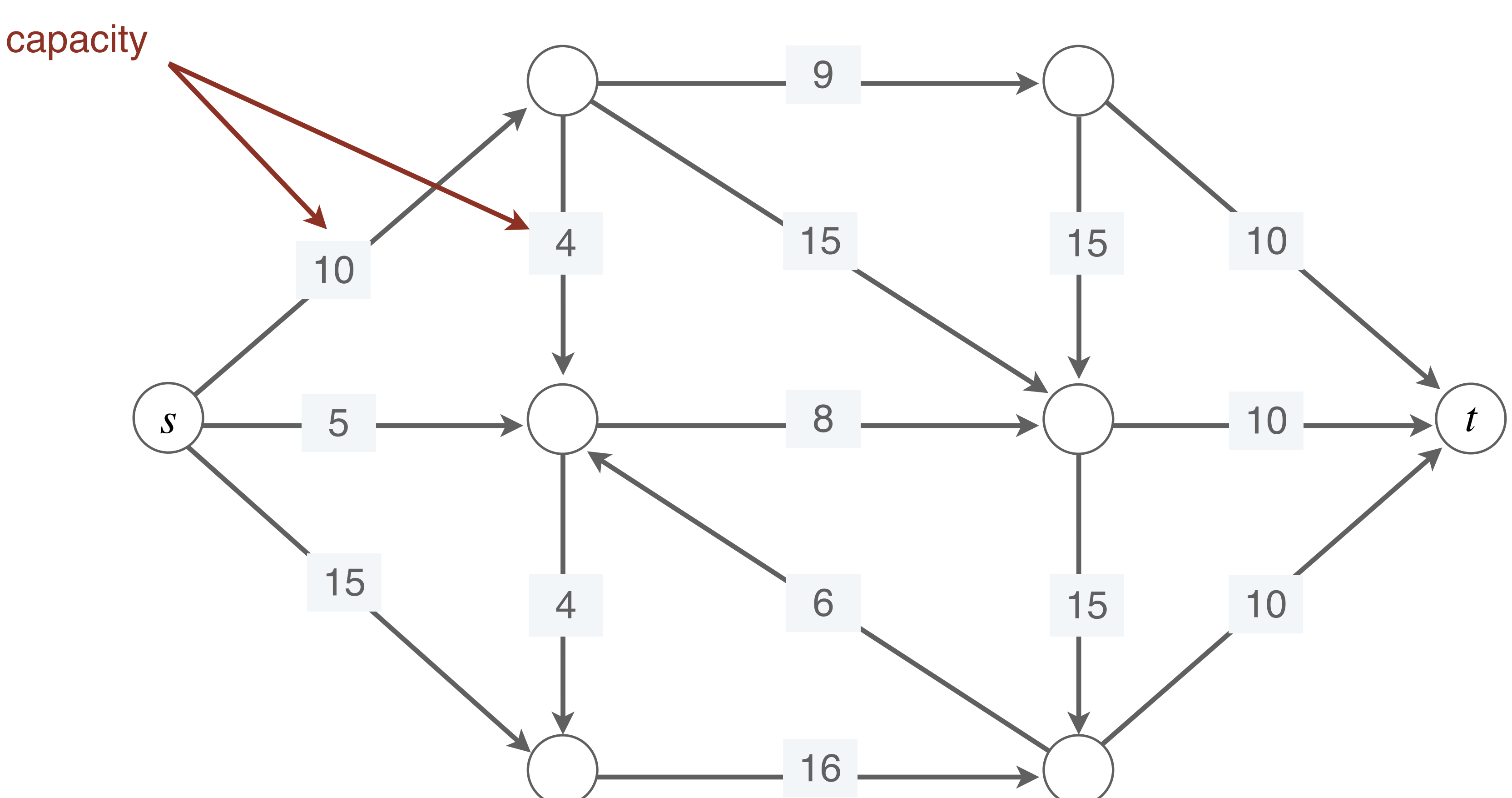


Image Credits: — Jeff Erickson's book and T[homas] E. Harris and F[rank] S. Ross. Fundamentals of a method for evaluating rail net capacities. The RAND Corporation, Research Memorandum RM-1517, October 24, 1955. United States Government work in the public domain. http://www.dtic.mil/dtic/tr/fulltext/u2/093458.pdf

What's a Flow Network?

- A flow network is a directed graph $G = (V, E)$ with a
  - A **source** is a vertex s with in degree 0
  - A **sink** is a vertex t with out degree 0
  - Each edge $e \in E$ has **edge capacity** $c(e) > 0$



Simplifying Assumptions

- Assume that each node v is on some s-t path, that is, $s \rightsquigarrow v \rightsquigarrow t$ exists, for any vertex $v \in V$
    - Implies G is connected, and $m \geq n - 1$
- Assume **capacities are integers**
- For simplifying expositions, assume $c(e) = 0$ if $e = (u, v)$ is not an edge, that is, for $u, v \in V$ and edge $(u, v) \notin E$
- Non-existent edges/capacities not shown in figures
- Directed edge (u, v) written as $u \to v$

What's a Flow?

- Given a flow network, an (s, t)-flow or just flow (if source s and sink t are clear from context) $f : E \rightarrow \mathbb{Z}^+$ satisfies:
- **[Flow conservation]** $f_{in}(v) = f_{out}(v)$, for $v \neq s, t$ where

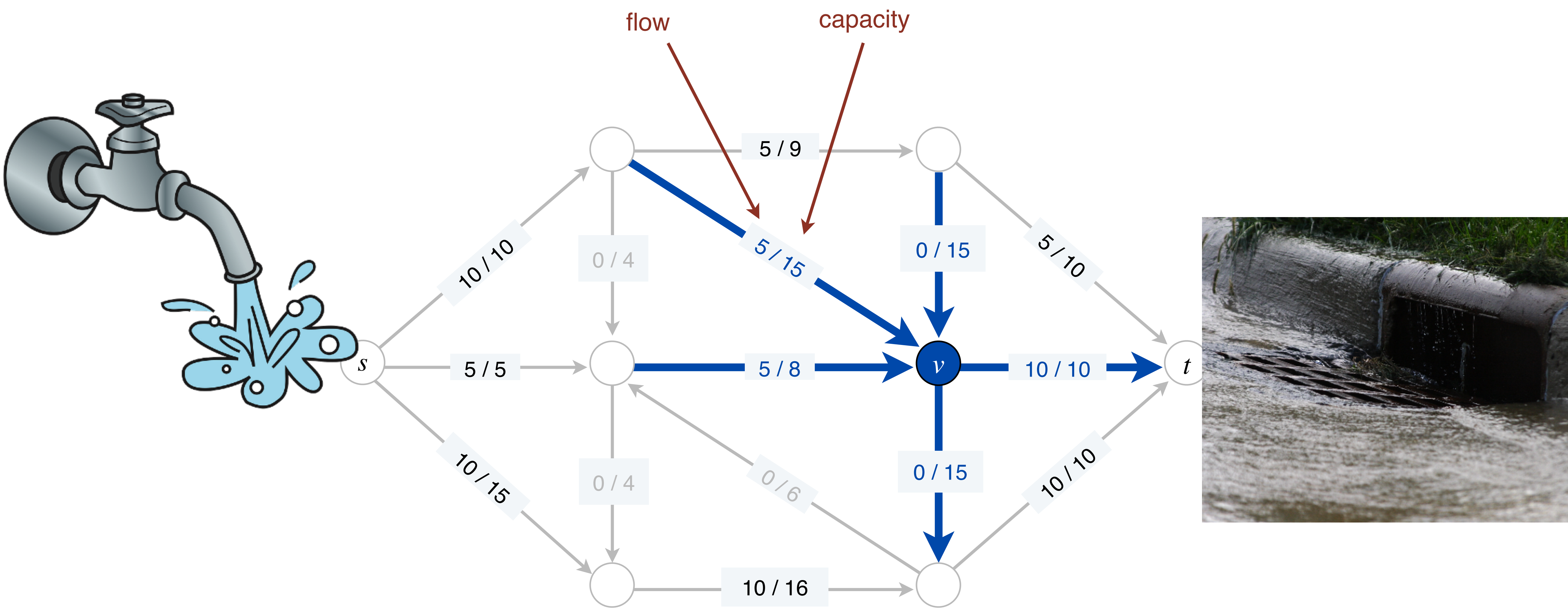
$$f_{in}(v) = \sum_u f(u \rightarrow v) \text{ and } f_{out}(v) = \sum_w f(v \rightarrow w)$$

- To simplify, $f(u \rightarrow v) = 0$ if there is no edge from u to v

What is a Feasible Flow

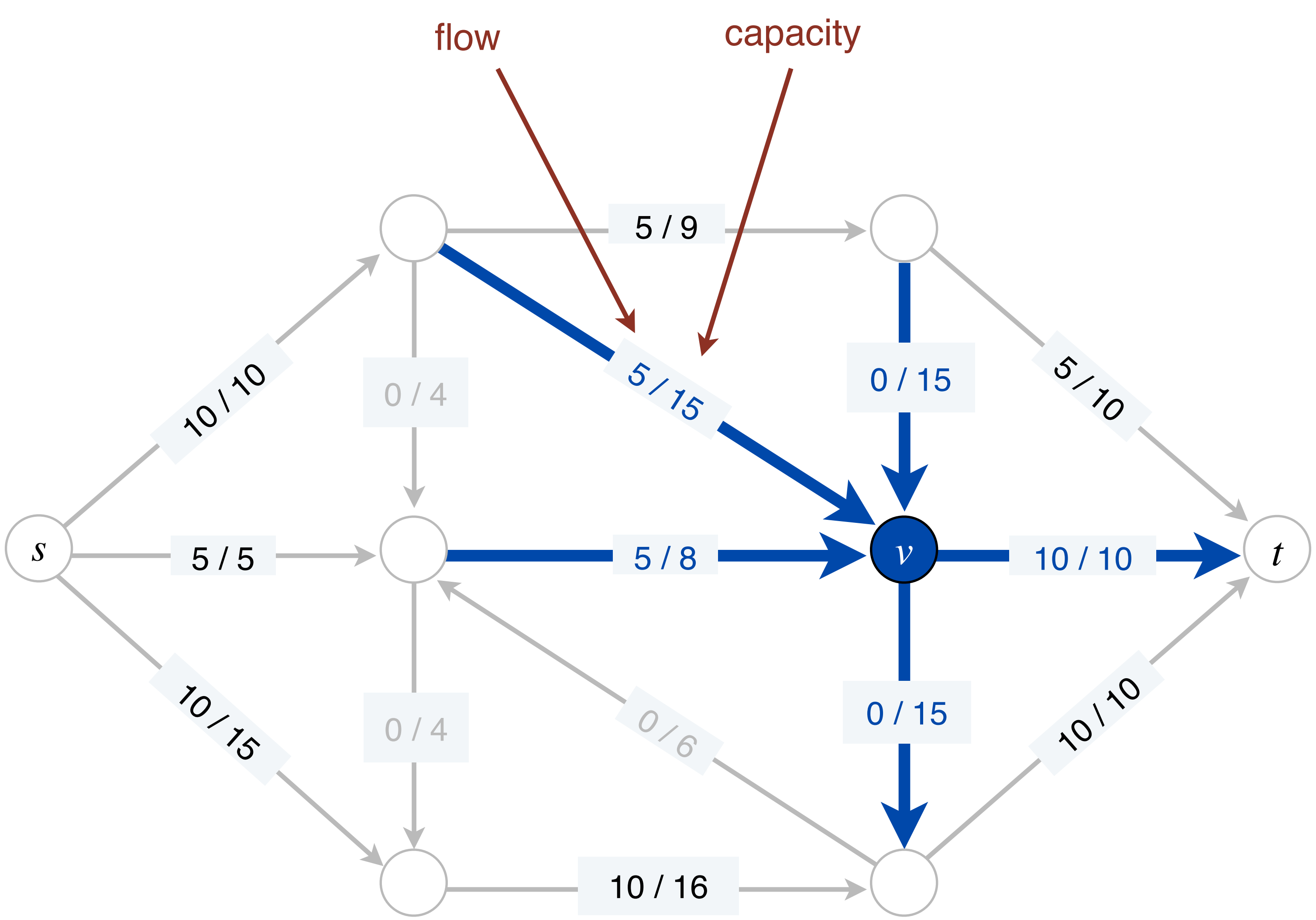
- An (s, t)-flow is **feasible** if it satisfies the capacity constraints of the network, that is,:

[Capacity constraint] for each $e \in E$, $0 \leq f(e) \leq c(e)$



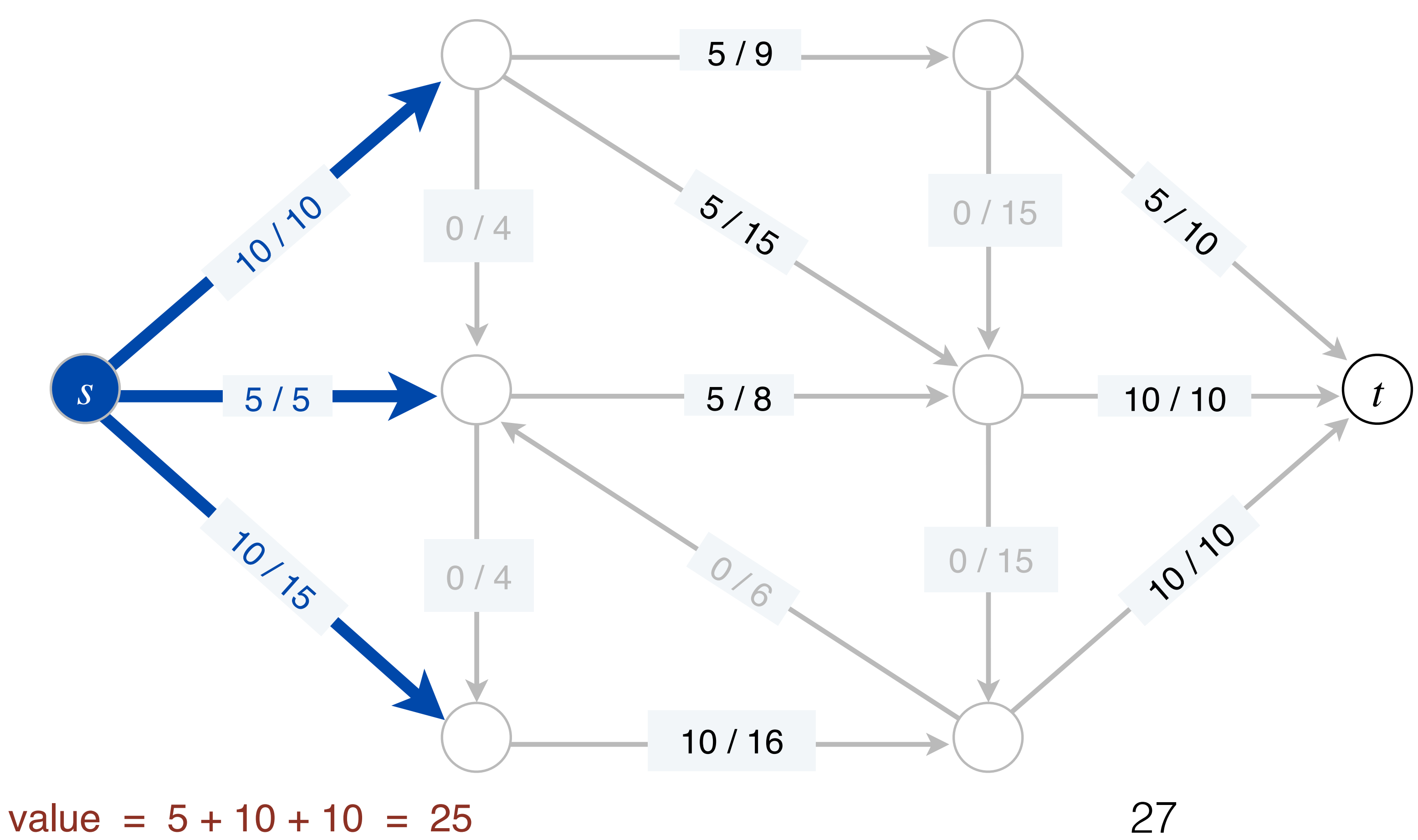
Value of a Flow

- **Definition.** The **value** of a flow f, written $v(f)$, is $f_{out}(s)$.



Value of a Flow

- **Definition.** The **value** of a flow f, written $v(f)$, is $f_{out}(s)$.
- **Lemma**. $f_{out}(s) = f_{in}(t)$



Value of a Flow

- **Definition.** The **value** of a flow f, written $v(f)$, is $f_{out}(s)$.
- **Lemma**. $f_{out}(s) = f_{in}(t)$
- **Proof**. Let $f(E) = \sum_{e \in E} f(e)$
    - Then, $\sum_{v \in V} f_{in}(v) = f(E) = \sum_{v \in V} f_{out}(v)$
    - For every $v \neq s, t$ flow conversation implies $f_{in}(v) = f_{out}(v)$
    - Thus all terms cancel out on both sides except $f_{in}(s) + f_{in}(t) = f_{out}(s) + f_{out}(t)$
    - But $f_{in}(s) = f_{out}(t) = 0$ ∎

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)