

# COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

## LECTURE 33

### NETWORK FLOW IV :

### EDMONDS - KARP ALGORITHM

OCT 25, 2024

|

ROHIT VAISH

# FORD - FULKERSON ALGORITHM



# FORD - FULKERSON ALGORITHM

① initialize  $f_e = 0 \quad \forall e$

② repeat :

\* search for an  $s-t$  path  $P$  in  $G_f$   
such that every edge in  $P$  has positive residual capacity

// possible in  $O(|E|)$  time via BFS/DFS

\* if no such path, return current flow  $\{f_e\}_{e \in E}$

\* else, let  $\Delta := \min_{e \in P} e$ 's residual capacity in  $G_f$

└ for all  $e \in G$  with forward edges in  $P$ , add  $\Delta$  to  $f_e$ .  
for all  $e \in G$  with reverse edges in  $P$ , subtract  $\Delta$  from  $f_e$ .

# FORD - FULKERSON ALGORITHM

For integral capacities,

Ford-Fulkerson terminates in finite time with a valid flow.

# FORD - FULKERSON ALGORITHM

For integral capacities,

Ford - Fulkerson terminates in finite time with a valid flow.

Does the algorithm return a max flow?

# FORD - FULKERSON ALGORITHM

For integral capacities,

Ford-Fulkerson terminates in finite time with a valid flow.

Does the algorithm return a max flow?



HOW DO WE KNOW WHEN WE ARE DONE?

# OPTIMALITY CONDITIONS

A general recipe :

- ① (Structure) Identify "optimality conditions".
- ② (Algorithm) Design an algorithm that terminates with the optimality conditions satisfied.

# OPTIMALITY CONDITIONS

A general recipe :

① (Structure) Identify "optimality conditions".

Claim : If  $G_f$  has no  $s-t$  path, then  $f$  is a max flow in  $G$

② (Algorithm) Design an algorithm that terminates with the optimality conditions satisfied.

Ford-Fulkerson algorithm

# OPTIMALITY CONDITIONS FOR MAX FLOW

**Theorem:** For a flow  $f$  in a graph  $G$ , the following are equivalent:

(a)  $f$  is a maximum flow of  $G$

(b) there exists an  $(s,t)$ -cut  $(A,B)$  such that  
value of  $f$  = capacity of  $(A,B)$ .

(c) there is no  $(s,t)$  path in the residual graph  $G_f$ .

# OPTIMALITY CONDITIONS FOR MAX FLOW

**Theorem:** For a flow  $f$  in a graph  $G$ , the following are equivalent:

(a)  $f$  is a maximum flow of  $G$ .

(b) there exists an  $(s,t)$ -cut  $(A,B)$  such that  
value of  $f$  = capacity of  $(A,B)$ .

(c) there is no  $(s,t)$  path in the residual graph  $G_f$ .

$(b) \implies (a)$



"TIGHT"  $(s,t)$ -CUT  $\Rightarrow$  MAX FLOW

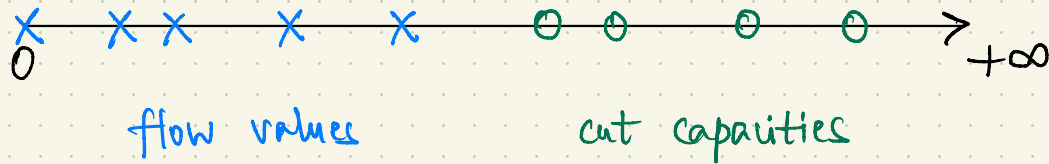
Sufficient to show that for every flow  $f$  and every  $(s,t)$ -cut  $(A,B)$ :

value of  $f \leq$  capacity of  $(A,B)$ .

# "TIGHT" (s,t)-CUT $\Rightarrow$ MAX FLOW

Sufficient to show that for every flow  $f$  and every (s,t)-cut  $(A,B)$ :

value of  $f \leq$  capacity of  $(A,B)$ .



# "TIGHT" (s,t)-CUT $\Rightarrow$ MAX FLOW

Sufficient to show that for every flow  $f$  and every (s,t)-cut  $(A,B)$ :

value of  $f \leq$  capacity of  $(A,B)$ .

Proof:

$$\text{value of } f = \sum_{v \in A} \left( \sum_{e \in \delta^+(v)} f_e - \sum_{e \in \delta^-(v)} f_e \right)$$

$$= \sum_{e \in \delta^+(A)} f_e - \underbrace{\sum_{e \in \delta^-(A)} f_e}_{\geq 0}$$

$$\leq \sum_{e \in \delta^+(A)} f_e \leq \sum_{e \in \delta^+(A)} u_e = \text{capacity of cut } (A,B). \quad \square$$

# OPTIMALITY CONDITIONS FOR MAX FLOW

**Theorem:** For a flow  $f$  in a graph  $G$ , the following are equivalent:

(a)  $f$  is a maximum flow of  $G$

(b) there exists an  $(s,t)$ -cut  $(A,B)$  such that  
value of  $f$  = capacity of  $(A,B)$ .

(c) there is no  $(s,t)$  path in the residual graph  $G_f$ .

(a)  $\implies$  (c)

# OPTIMALITY CONDITIONS FOR MAX FLOW

**Theorem:** For a flow  $f$  in a graph  $G$ , the following are equivalent:

(a)  $f$  is a maximum flow of  $G$

(b) there exists an  $(s,t)$ -cut  $(A,B)$  such that  
value of  $f$  = capacity of  $(A,B)$ .

(c) there is no  $(s,t)$  path in the residual graph  $G_f$ .

(a)  $\implies$  (c)

Proof by Ford Fulkerson

# OPTIMALITY CONDITIONS FOR MAX FLOW

**Theorem:** For a flow  $f$  in a graph  $G$ , the following are equivalent:

(a)  $f$  is a maximum flow of  $G$

(b) there exists an  $(s,t)$ -cut  $(A,B)$  such that  
value of  $f$  = capacity of  $(A,B)$ .

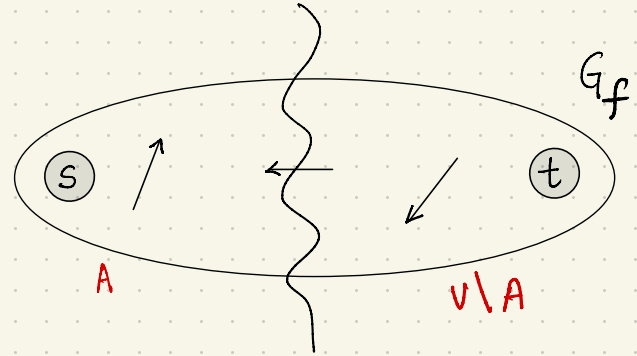
(c) there is no  $(s,t)$  path in the residual graph  $G_f$ .

(c)  $\Rightarrow$  (b)

No  $(s,t)$  PATH IN  $G_f \Rightarrow$  "TIGHT" CUT

Define  $A := \{ v \in V : \text{there is an s-w-v path in } G_f \}$ .

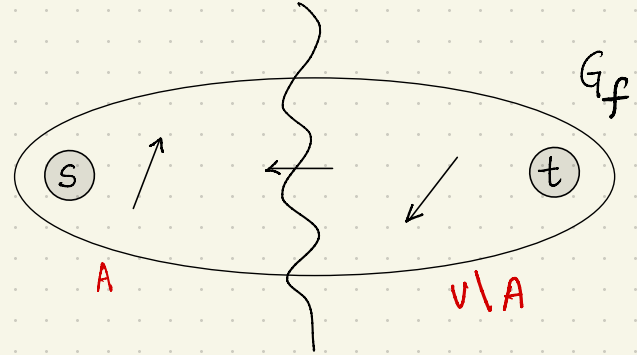
Then,  $(A, V \setminus A)$  is an  $(s,t)$ -cut.



No  $(s,t)$  PATH IN  $G_f \Rightarrow$  "TIGHT" CUT

Define  $A := \{ v \in V : \text{there is an s-m-v path in } G_f \}$ .

Then,  $(A, V \setminus A)$  is an  $(s,t)$ -cut.



In the underlying graph  $G$ :

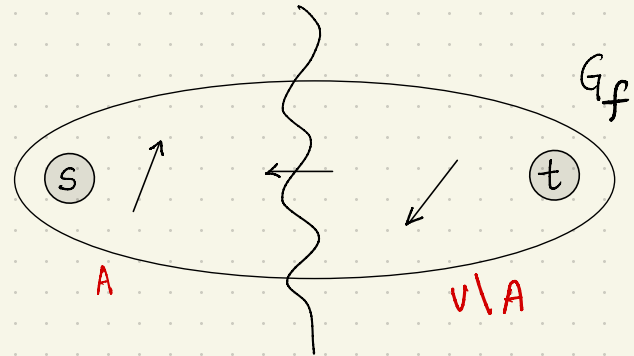
- \* Every edge  $e \in \delta^+(A)$  is **saturated** (i.e.,  $u_e = f_e$ ).
- \* Every edge  $e \in \delta^-(A)$  is **zeroed out** (i.e.,  $f_e = 0$ ).



No  $(s,t)$  PATH IN  $G_f \Rightarrow$  "TIGHT" CUT

Define  $A := \{ v \in V : \text{there is an srrv path in } G_f \}$ .

Then,  $(A, V \setminus A)$  is an  $(s,t)$ -cut.



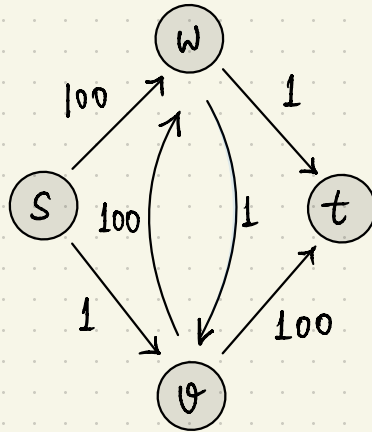
In the underlying graph  $G$ :

- \* Every edge  $e \in \delta^+(A)$  is **saturated** (i.e.,  $u_e = f_e$ ).
- \* Every edge  $e \in \delta^-(A)$  is **zeroed out** (i.e.,  $f_e = 0$ ).

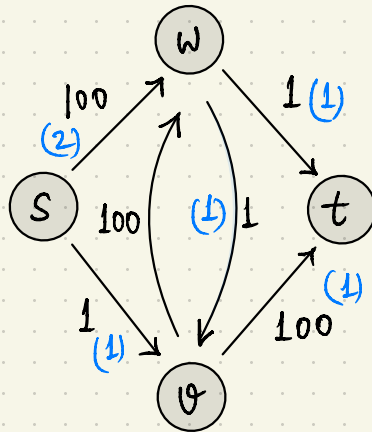
$\Rightarrow (A, V \setminus A)$  is a tight cut.



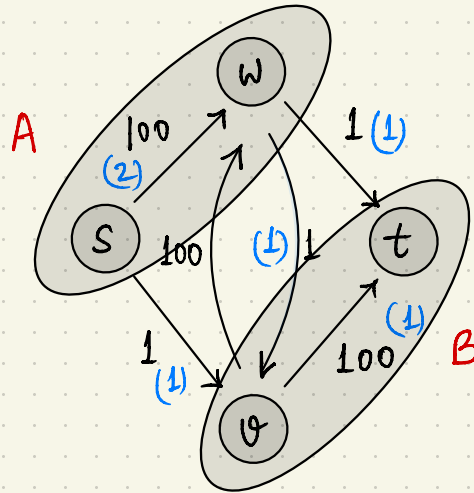
# OPTIMALITY CONDITIONS FOR MAX FLOW



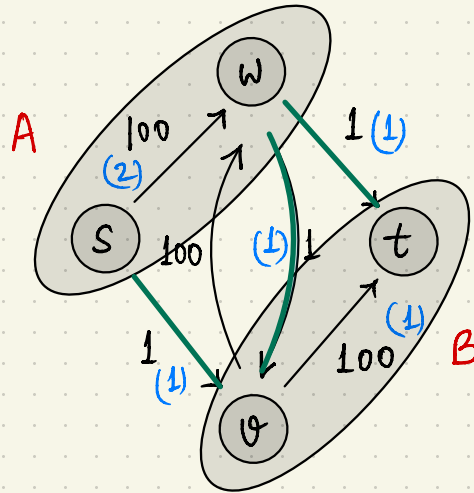
# OPTIMALITY CONDITIONS FOR MAX FLOW



# OPTIMALITY CONDITIONS FOR MAX FLOW

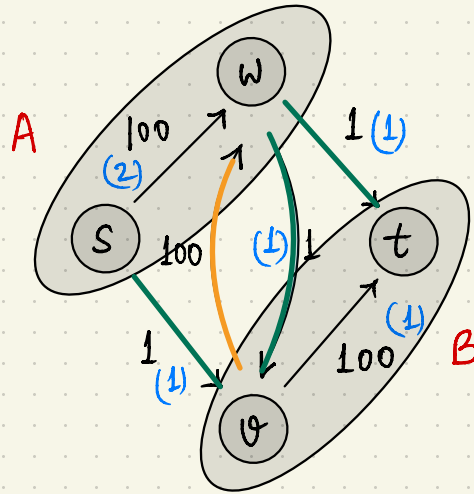


# OPTIMALITY CONDITIONS FOR MAX FLOW



$\delta^+(A)$ : saturated

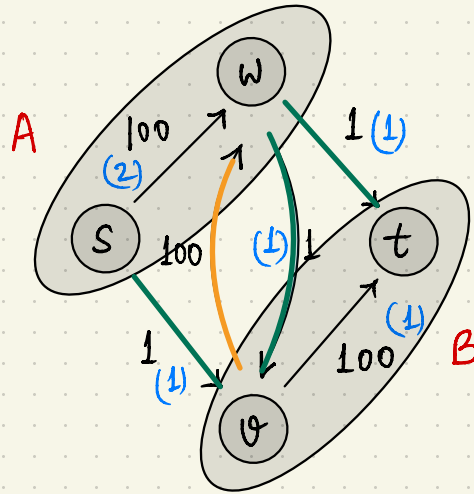
# OPTIMALITY CONDITIONS FOR MAX FLOW



$\delta^+(A)$ : saturated

$\delta^-(A)$ : zeroed out

# OPTIMALITY CONDITIONS FOR MAX FLOW

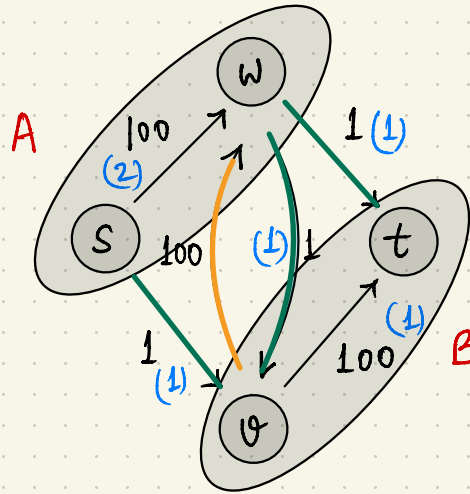


$\delta^+(A)$ : saturated

$\delta^-(A)$ : zeroed out

$\Rightarrow (A, B)$  is a tight cut

# OPTIMALITY CONDITIONS FOR MAX FLOW



$\delta^+(A)$ : saturated

$\delta^-(A)$ : zeroed out

$\Rightarrow (A, B)$  is a tight cut

$\Rightarrow$  max flow



# MAX-FLOW MIN-CUT THEOREM

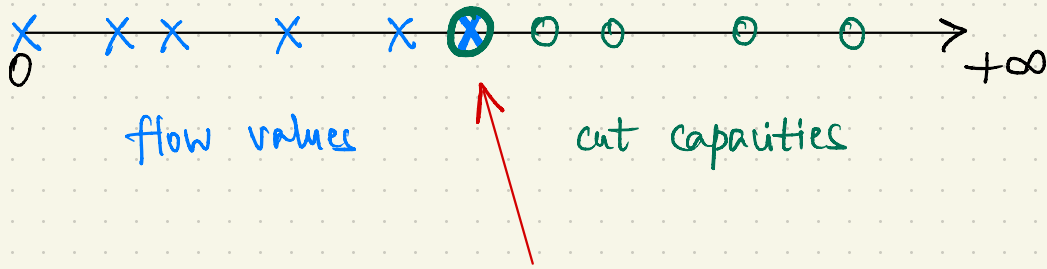
Corollary : For every network,

maximum value of a flow = minimum capacity of an  $(s,t)$ -cut

# MAX-FLOW MIN-CUT THEOREM

Corollary: For every network,

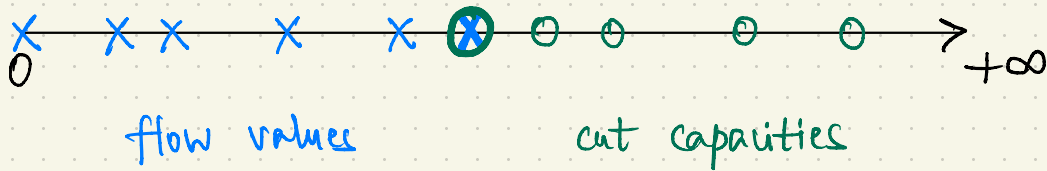
maximum value of a flow = minimum capacity of an  $(s,t)$ -cut



# MAX-FLOW MIN-CUT THEOREM

Corollary: For every network,

maximum value of a flow = minimum capacity of an  $(s,t)$ -cut

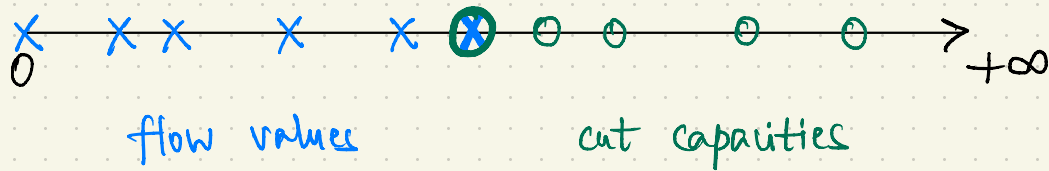


Corollary (of Corollary): Given a maximum flow, a minimum cut can be recovered in  $O(|E|)$  time.

# MAX-FLOW MIN-CUT THEOREM

Corollary: For every network,

maximum value of a flow = minimum capacity of an  $(s,t)$ -cut



Corollary (of Corollary): Given a maximum flow, a minimum cut can be recovered in  $O(|E|)$  time.

Proof: Do BFS/DFS from  $s$ , construct the set  $A$  of reachable vertices. Then,  $(A, V \setminus A)$  is a min cut. □

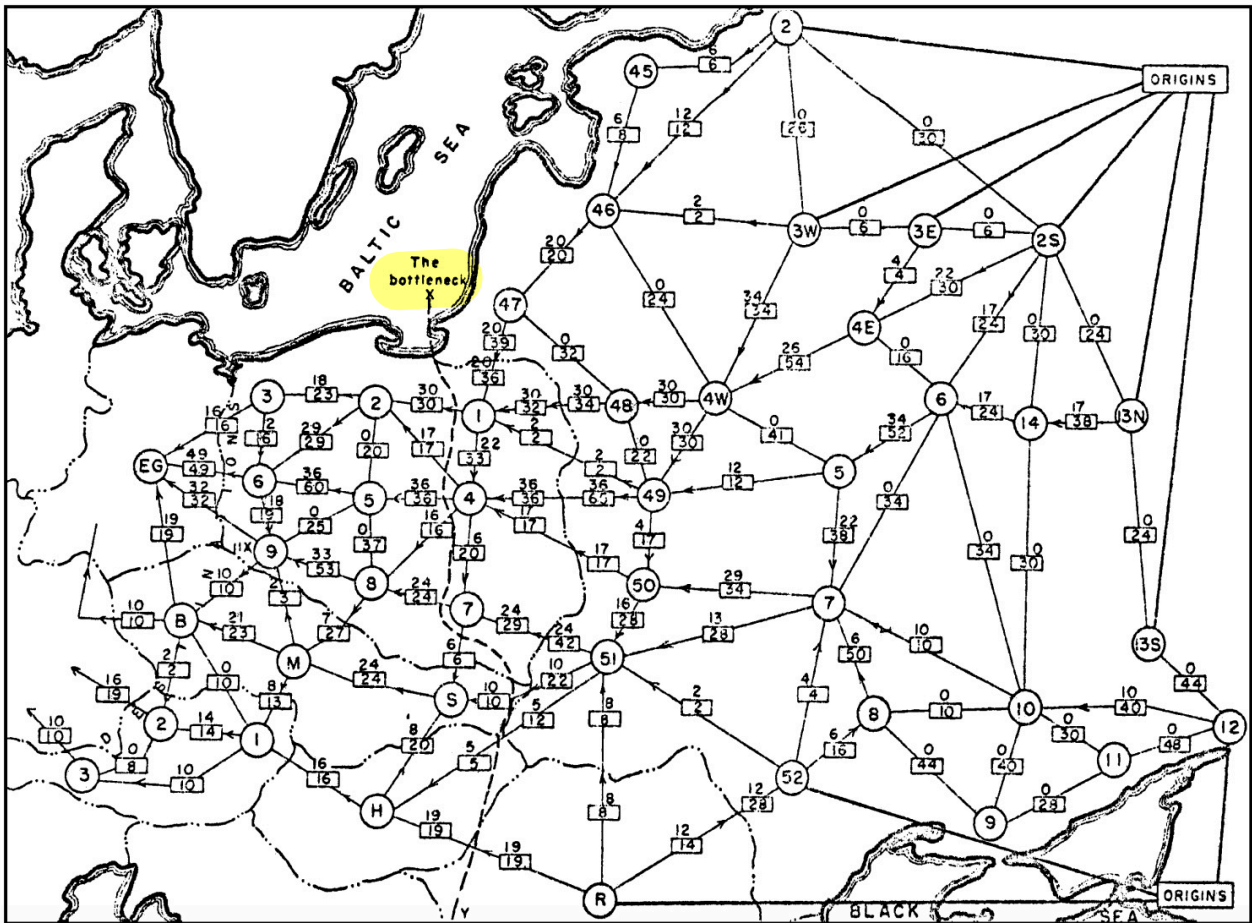
# MAX-FLOW MIN-CUT THEOREM

**On the history of combinatorial optimization  
(till 1960)**

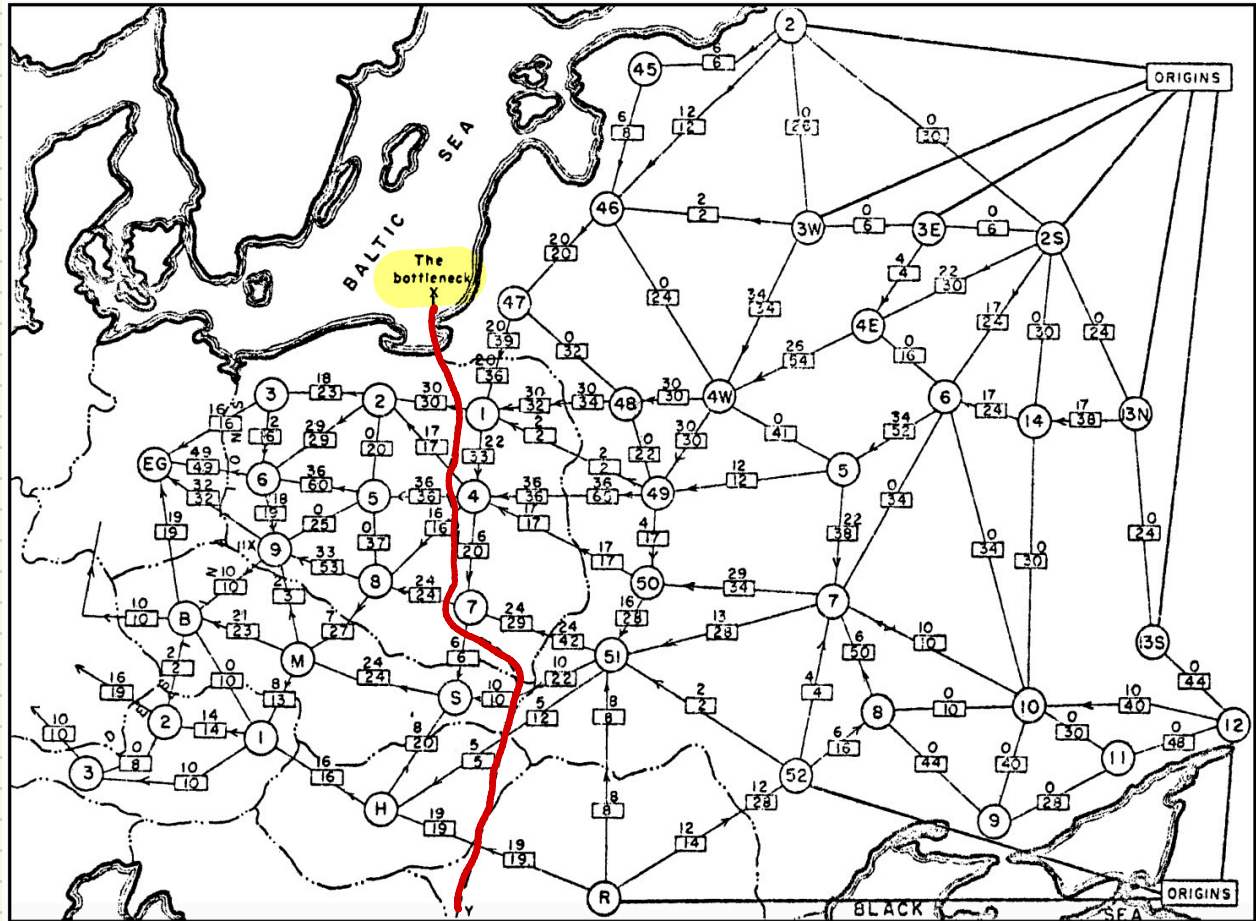
Alexander Schrijver<sup>1</sup>



# MAX-FLOW MIN-CUT THEOREM



# MAX-FLOW MIN-CUT THEOREM





BACK TO FORD - FULKERSON

## BACK TO FORD-FULKERSON

\* Recall :  $f$  is max flow  $\iff$  no  $(s,t)$ -path in  $G_f$

$\implies$  Ford-Fulkerson is **correct**

## BACK TO FORD-FULKERSON

\* Recall :  $f$  is max flow  $\iff$  no  $(s,t)$ -path in  $G_f$

$\implies$  Ford-Fulkerson is correct

\* Running time of Ford-Fulkerson : Pseudopolynomial (Tutorial 11)

runtime depends on  $U_e$  (not  $\log U_e$ )

# BACK TO FORD-FULKERSON

\* Recall :  $f$  is max flow  $\iff$  no  $(s,t)$ -path in  $G_f$

$\implies$  Ford-Fulkerson is correct

\* Running time of Ford-Fulkerson : Pseudopolynomial (Tutorial 11)

runtime depends on  $U_e$  (not  $\log U_e$ )

Keeps choosing "essentially useless" paths over and over

# BACK TO FORD-FULKERSON

\* Recall :  $f$  is max flow  $\iff$  no  $(s,t)$ -path in  $G_f$

$\implies$  Ford-Fulkerson is correct

\* Running time of Ford-Fulkerson : Pseudopolynomial (Tutorial 11)

runtime depends on  $U_e$  (not  $\log U_e$ )

Keeps choosing "essentially useless" paths over and over



Choose  $s-t$  paths intelligently

# BACK TO FORD-FULKERSON

\* Recall :  $f$  is max flow  $\iff$  no  $(s,t)$ -path in  $G_f$

$\implies$  Ford-Fulkerson is correct

\* Running time of Ford-Fulkerson : Pseudopolynomial (Tutorial 11)

runtime depends on  $U_e$  (not  $\log U_e$ )

\* Fix : Edmonds-Karp algorithm

# EDMONDS-KARP ALGORITHM

# EDMONDS-KARP ALGORITHM

Refinement of Ford-Fulkerson

In  $G_f$ , pick an  $s-t$  path with **fewest** number of edges.



# EDMONDS-KARP ALGORITHM

Ford-  
Fulkerson

① initialize  $f_e = 0 \quad \forall e$

② repeat:

\* search for an  $s-t$  path  $P$  in  $G_f$

such that every edge in  $P$  has positive residual capacity

// possible in  $O(|E|)$  time via BFS/DFS

\* if no such path, return current flow  $\{f_e\}_{e \in E}$

\* else, let  $\Delta := \min_{e \in P} e$ 's residual capacity in  $G_f$

└ for all  $e \in E$  with forward edges in  $P$ , add  $\Delta$  to  $f_e$ .

for all  $e \in E$  with reverse edges in  $P$ , subtract  $\Delta$  from  $f_e$ .

# EDMONDS-KARP ALGORITHM

① initialize  $f_e = 0 \quad \forall e$

② repeat:

\* search for an  $s-t$  path  $P$  in  $G_f$

such that every edge in  $P$  has positive residual capacity

// possible in  $O(|E|)$  time via BFS/DFS

\* if no such path, return current flow  $\{f_e\}_{e \in E}$

\* else, let  $\Delta := \min_{e \in P} e$ 's residual capacity in  $G_f$

└ for all  $e \in E$  with forward edges in  $P$ , add  $\Delta$  to  $f_e$ .

for all  $e \in E$  with reverse edges in  $P$ , subtract  $\Delta$  from  $f_e$ .

# EDMONDS-KARP ALGORITHM

① initialize  $f_e = 0 \quad \forall e$

② repeat:

\* search for an  $s-t$  path  $P$  in  $G_f$  with the fewest number of edges such that every edge in  $P$  has positive residual capacity

// possible in  $O(|E|)$  time via BFS/DFS

\* if no such path, return current flow  $\{f_e\}_{e \in E}$

\* else, let  $\Delta := \min_{e \in P} e$ 's residual capacity in  $G_f$

└ for all  $e \in G$  with forward edges in  $P$ , add  $\Delta$  to  $f_e$ .

for all  $e \in G$  with reverse edges in  $P$ , subtract  $\Delta$  from  $f_e$ .

# EDMONDS-KARP ALGORITHM

① initialize  $f_e = 0 \quad \forall e$

② repeat:

\* search for an  $s-t$  path  $P$  in  $G_f$  with fewest no. of edges such that every edge in  $P$  has positive residual capacity

// possible in  $O(|E|)$  time via BFS/DFS

\* if no such path, return current flow  $\{f_e\}_{e \in E}$

\* else, let  $\Delta := \min_{e \in P} e$ 's residual capacity in  $G_f$

└ for all  $e \in G$  with forward edges in  $P$ , add  $\Delta$  to  $f_e$ .

for all  $e \in G$  with reverse edges in  $P$ , subtract  $\Delta$  from  $f_e$ .

# EDMONDS-KARP ALGORITHM

\* *Correct* (special case of Ford - Fulkerson)

# EDMONDS-KARP ALGORITHM

\* **Correct** (special case of Ford - Fulkerson)

**Theorem:** The Edmonds-Karp algorithm runs in  $O(m^2n)$  time,  
where  $m = |E|$  and  $n = |V|$ .

# EDMONDS-KARP ALGORITHM

\* **Correct** (special case of Ford - Fulkerson)

**Theorem**: The Edmonds-Karp algorithm runs in  $O(m^2n)$  time,  
where  $m = |E|$  and  $n = |V|$ .

\* Polynomial-time for any input capacities  $\{u_e\}$ .

# EDMONDS-KARP ALGORITHM

\* **Correct** (special case of Ford - Fulkerson)

**Theorem**: The Edmonds-Karp algorithm runs in  $O(m^2n)$  time,  
where  $m = |E|$  and  $n = |V|$ .

\* Polynomial-time for any input capacities  $\{c_e\}$ .

\*  $O(n^3)$  for sparse graphs,  $O(n^5)$  for dense graphs



# RUNNING TIME ANALYSIS OF EDMONDS-KARP

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Recall:

$f$  is max flow  $\iff$  no  $(s,t)$  path in  $G_f$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Recall:

$f$  is max flow  $\iff$  no  $(s,t)$  path in  $G_f$


want to disconnect  
 $s$  and  $t$  in  $G_f$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Recall:

$f$  is max flow  $\iff$  no  $(s,t)$  path in  $G_f$

want to disconnect  
 $s$  and  $t$  in  $G_f$


 Measure progress in terms of **how far**  $s$  and  $t$  are in  $G_f$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Recall:

$f$  is max flow  $\iff$  no  $(s,t)$  path in  $G_f$

want to **disconnect**  
 $s$  and  $t$  in  $G_f$

 Measure progress in terms of **how far**  $s$  and  $t$  are in  $G_f$


number of edges  
on the minimum-edge path

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Recall:

$f$  is max flow  $\iff$  no  $(s,t)$  path in  $G_f$

want to disconnect  
 $s$  and  $t$  in  $G_f$

 Measure progress in terms of **how far**  $s$  and  $t$  are in  $G_f$

**NOTE:** An  $(s,t)$  path can have at most  $(n-1)$  edges.

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ .

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \left[ \right.$$



# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \left[ \begin{array}{l} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \end{array} \right.$$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \quad \text{if no such path exists.} \end{cases}$$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

Then,

(a)  $d(f)$  never decreases during the execution of the algorithm

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

**Key lemma:** Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

- Then,
- (a)  $d(f)$  never decreases during the execution of the algorithm
  - (b)  $d(f)$  increases at least once per  $m$  iterations.

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

**Key lemma:** Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

Then,

(a)  $d(f)$  never decreases during the execution of the algorithm

(b)  $d(f)$  increases at least once per  $m$  iterations.

$G$  fixed,  $f$  changes  $\Rightarrow G_f$  changes

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

**Key lemma:** Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

- Then,
- (a)  $d(f)$  never decreases during the execution of the algorithm
  - (b)  $d(f)$  increases at least once per  $m$  iterations.

$$d(f) \in \{0, 1, \dots, n-1, +\infty\}$$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

- Then,
- (a)  $d(f)$  never decreases during the execution of the algorithm
  - (b)  $d(f)$  increases at least once per  $m$  iterations.

$$d(f) \in \{0, 1, \dots, n-1, +\infty\} \quad \text{Once } d(f) \geq n \Rightarrow d(f) = +\infty$$

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

- Then,
- (a)  $d(f)$  never decreases during the execution of the algorithm
  - (b)  $d(f)$  increases at least once per  $m$  iterations.

$$d(f) \in \{0, 1, \dots, n-1, +\infty\} \quad \text{Once } d(f) \geq n \Rightarrow d(f) = +\infty \text{ done!}$$



# RUNNING TIME ANALYSIS OF EDMONDS-KARP

**Key lemma:** Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

- Then,
- (a)  $d(f)$  never decreases during the execution of the algorithm
  - (b)  $d(f)$  increases at least once per  $m$  iterations.

$$d(f) \in \{0, 1, \dots, n-1, +\infty\}$$

Once  $d(f) \geq n \Rightarrow d(f) = +\infty$  done!

$O(mn)$  iterations

# RUNNING TIME ANALYSIS OF EDMONDS-KARP

Key lemma: Fix a network  $G$ . For a flow  $f$ , define:

$$d(f) = \begin{cases} \text{the number of edges in a shortest } s-t \text{ path in } G_f \\ \text{with positive residual capacity} \\ +\infty \text{ if no such path exists.} \end{cases}$$

- Then,
- (a)  $d(f)$  never decreases during the execution of the algorithm
  - (b)  $d(f)$  increases at least once per  $m$  iterations.

$$d(f) \in \{0, 1, \dots, n-1, +\infty\}$$

Once  $d(f) \geq n \Rightarrow d(f) = +\infty$  done!  
 $O(mn)$  iterations  $\xrightarrow{\text{BFS per iteration}}$   $O(m^2n)$  time.