

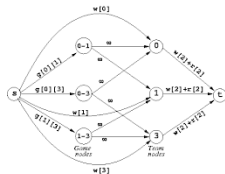
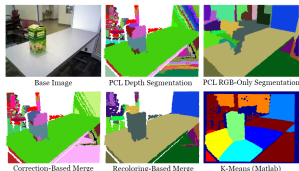
# Max-flow and min-cut

# Max-Flow and Min-Cut

Two important algorithmic problems, which yield a beautiful duality

Myriad of non-trivial applications, it plays an important role in the optimization of many problems:

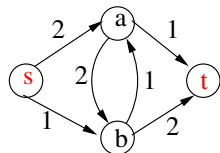
Network connectivity, airline schedule (extended to all means of transportation), image segmentation, bipartite matching, distributed computing, data mining, . . . . .



# Flow Networks

Network diagram  $G = (V, E)$  s.t. it has

- ▶ source vertex  $s \in V$
- ▶ sink vertex  $t \in V$
- ▶ edge capacities  $c : E \rightarrow \mathbb{R}^+ \cup \{0\}$

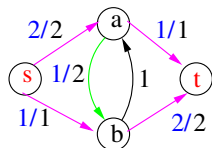


Flow  $f : V \times V \rightarrow \mathbb{R}^+ \cup \{0\}$  s.t.

**Kirchoff's laws:**

- ▶  $\forall (u, v) \in E, 0 \leq f(u, v) \leq c(u, v),$
- ▶ (Flow conservation)  $\forall v \in V - \{s, t\},$   
 $\sum_{u \in V} f(u, v) = \sum_{z \in V} f(v, z)$
- ▶ The **value of a flow**

$$|f| = \sum_{v \in V} f(s, v) = f(s, V) = f(V, t).$$

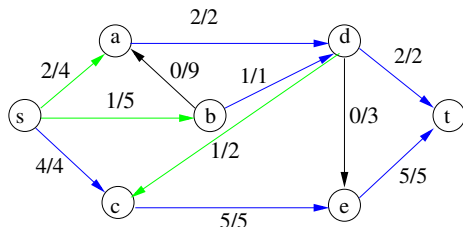


**Value  $|f|=3$**

# The Maximum flow problem

INPUT: Given a flow network  $(G = (V, E), s, t, c)$

QUESTION: Find a flow of maximum value on  $G$ .



The value of the max-flow is  $7 = 4 + 1 + 2 = 5 + 2$ .

Notice: Although the flow exiting  $s$  is not maximum, the flow going into  $t$  is maximum (= max. capacity).

Therefore the total flow is maximum.

## The $s - t$ cut

Given  $(G = (V, E), s, t, c)$  a  $s - t$  cut is a partition of  $V = S \cup T$  ( $S \cap T = \emptyset$ ), with  $s \in S$  and  $t \in T$ .

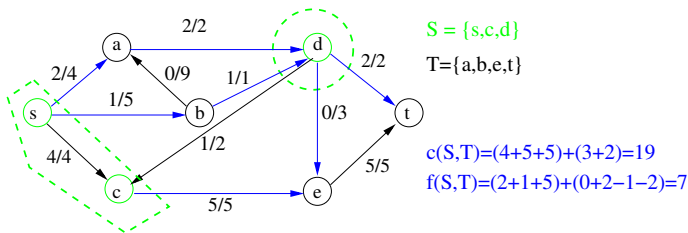
The flow across the cut:

$$f(S) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{v \in S} \sum_{u \in T} f(v, u).$$

The capacity of the cut:  $c(S) = \sum_{u \in S} \sum_{v \in T} c(u, v)$

capacity of cut  $(S, T) =$  sum of weights leaving  $S$ .

Notice because of the capacity constrain:  $f(S) \leq c(S)$



## The $s - t$ cut

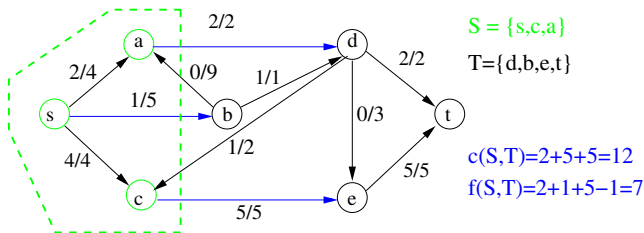
Given  $(G = (V, E), s, t, c)$  a  $s - t$  cut is a partition of  $S, T$  of  $V$  (i.e.  $V = S \cup T$  and  $S \cap T = \emptyset$ ), with  $s \in S$  and  $t \in T$ .

The flow across the cut:

$$f(S) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{v \in T} \sum_{u \in S} f(v, u).$$

The capacity of the cut:  $c(S) = \sum_{u \in S} \sum_{v \in T} c(u, v)$

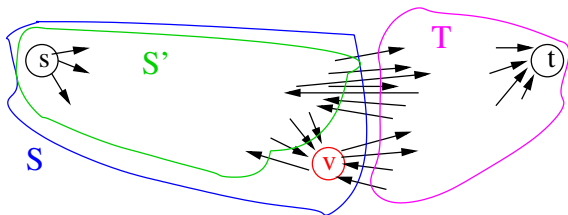
Notice because of the capacity constrain:  $f(S) \leq c(S)$



# Notation

Given  $v \in G$  and cut  $(S, T)$  and a  $v \in S$ , let  $S' = S - \{v\}$ . Then

- ▶ Denote  $f(S', T)$  flow between  $S'$  and  $T$  (without going by  $v$ ).  
i.e.  $f(S', T) = \sum_{u \in S'} \sum_{w \in T} f(u, w) - \sum_{w \in T} \sum_{u \in S'} f(w, u)$  with  
 $(u, w) \in E$  and  $(u, w) \in E$ ,
- ▶ denote  $f(v, T)$  flow  $v \rightarrow T$  i.e.  $f(v, T) = \sum_{u \in T} f(v, u)$ ,
- ▶ denote  $f(T, v)$  flow  $T \rightarrow v$  i.e.  $f(T, v) = \sum_{u \in T} f(u, v)$ ,
- ▶ denote  $f(S', v)$  flow  $S' \rightarrow v$  i.e.  $f(S', v) = \sum_{u \in S'} f(u, v)$ ,
- ▶ denote  $f(v, S')$  flow  $v \rightarrow S'$  i.e.  $f(v, S') = \sum_{u \in S'} f(v, u)$ ,



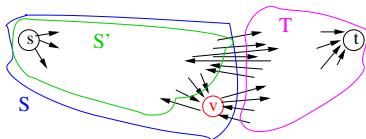
# Any $s - t$ cut has the same flow

## Theorem

Given  $(G, s, t, c)$  the flow through any  $s - t$  cut  $(S, T)$  is  $f(S) = |f|$ .

**Proof** (Induction on  $|S|$ )

- ▶ If  $S = \{s\}$  then  $f(S) = |f|$ .
- ▶ Assume it is true for  $S' = S - \{v\}$ , i.e.  $f(S') = |f|$ .  
Notice  $f(S') = f(S', T) + f(S', v) - f(v, S')$ . Moreover from the flow conservation,  $f(S', v) + f(T, v) = f(v, S') + f(v, T)$   
 $\Rightarrow \underbrace{f(v, T) - f(T, v)}_{*} = f(S', v) - f(v, S')$
- ▶ Then  $f(S) = f(S', T) + f(v, T) - f(T, v)$ , using (\*)  
 $f(S) = f(S') = |f|$  □

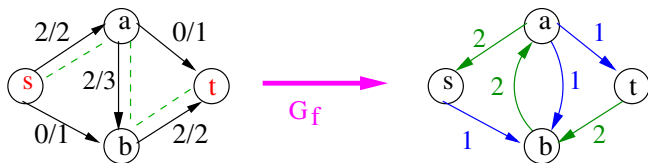




# Residual network

Given a network  $(G = (V, E), s, t, c)$  together with a flow  $f$  on it, the residual network,  $(G_f = (V, E_f), c_f)$  is the network with the same vertex set and edge set:

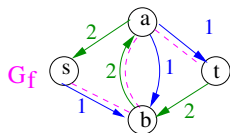
- ▶ if  $c(u, v) - f(u, v) > 0$  then  $(u, v) \in E_f$  and  $c_f(u, v) = c(u, v) - f(u, v) > 0$  (forward edges), and
- ▶ if  $f(u, v) > 0$  then  $(v, u) \in E_f$  and  $c_f(v, u) = f(u, v)$  (backward edges). i.e. there are  $f(u, v)$  units of flow we can undo, by pushing flow backward. Notice, if  $c(u, v) = f(u, v)$  then there is only a backward edge.
- ▶ the  $c_f$  are denoted residual capacity.



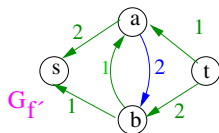
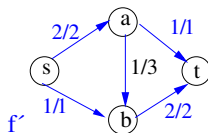
## Residual network: Augmenting paths

Given  $G = (V, E)$  and a flow  $f$  on  $G$ , an **augmenting path**  $P$  is any **simple** path in  $G_f$  (using forward and backward edges, but  $P : s \rightsquigarrow t$ ).

Given  $f : s \rightsquigarrow t$  in  $G$  and  $P$  in  $G_f$  define the **bottleneck**  $(P, f)$  to be the minimum residual capacity of any edge in  $P$ , with respect to  $f$ .



$P$ : dotted line



## Residual network: Augmenting paths

Given  $G = (V, E)$  and a flow  $f$  on  $G$ , an **augmenting path**  $P$  is any **simple** path in  $G_f$ .

Given  $f$   $s \rightarrow t$  in  $G$  and  $P$  in  $G_f$  define the **bottleneck**  $(P, f)$  to be the **minimum residual capacity** of any edge in  $P$ .

**Augment** $(P, f)$

$b = \text{bottleneck}(P, f)$

**for** each  $(u, v) \in P$  **do**

**if**  $(u, v)$  is forward edge in  $G$  **then**

    Increase  $f(u, v)$  in  $G$  by  $b$

**else**

    Decrease  $f(u, v)$  in  $G$  by  $b$

**end if**

**end for**

**return**  $f$

# Residual network: Augmenting paths

## Lemma

Consider  $f' = \text{Augment}(P, f)$ , then  $f'$  is a flow in  $G$ .

**Proof:** We have to prove that (1)  $\forall e \in E, 0 \leq f(e) \leq c(e)$  and that  $\forall v$  flow to  $v =$  flow out of  $v$ .

- ▶ **Capacity law** Forward edges  $(u, v) \in P$  we increase  $f(u, v)$  by  $b$ , as  $b \leq c(u, v) - f(u, v)$  then  $f'(u, v) = f(u, v) + b \leq c(u, v)$ .  
Backward edges  $(u, v) \in P$  we decrease  $f(v, u)$  by  $b$ , as  $b \leq f(v, u)$ ,  $f'(v, u) = f(v, u) - b \geq 0$ .
- ▶ **Conservation law**,  $\forall v \in P$  given edges  $e_1, e_2$  in  $P$  and incident to  $v$ , it is easy to check the 4 cases based whether  $e_1, e_2$  are forward or backward edges. □

# Max-Flow Min-Cut theorem

## Theorem

For any  $(G, s, t, c)$  the value of the max flow  $f^*$  is equal to the capacity of the min  $(S, T)$ -cut (over all  $s - t$  cuts in  $G$ )

$$f^* = \max\{|f|\} = \min_{\forall(S,T)} \{c(S, T)\}.$$

## Proof:

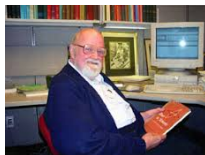
- ▶ For any  $s - t$  cut  $(S, T)$  in  $G \Rightarrow f^*(S) \leq c(S, T)$ .
- ▶ If  $f^*$  in  $G$  is a max flow then  $G_{f^*}$  has no augmenting path  $s \rightsquigarrow t$  so it is disconnected.

Let  $S_s = \{v \in V \mid \exists s \rightsquigarrow v \text{ in } G_{f^*}\}$ , then  $(S_s, V - \{S_s\})$  is a  $s - t$  cut in  $G_{f^*} \Rightarrow \forall v \in S_s, u \in V - \{S_s\}, (v, \vec{u})$  is not a residual edges, so in  $G$   $f^*(v, u) = c(v, u)$ , i.e.

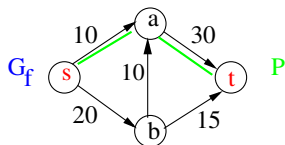
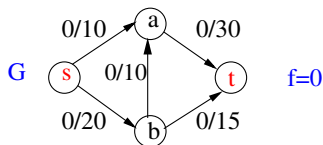
$c(S_s, V - \{S_s\}) = f^*(S_s, V - \{S_s\})$  in  $G$ . In particular  $(S_s, V - \{S_s\})$  is a min-cut in  $G$  and = max-flow  $f^*$ . □

# Ford-Fulkerson algorithm

L.R. Ford, D.R. Fulkerson:  
*Maximal flow through a network*. Canadian J. of Math. 1956.

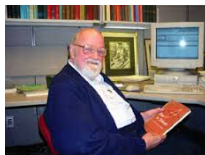


**Ford-Fulkerson**( $G, s, t, c$ )  
for all  $(u, v) \in E$  let  $f(u, v) = 0$   
 $G_f = G$   
**while** there is an  $s - t$  path in  $G_f$  **do**  
    find a simple path  $P$  in  $G_f$  (use DFS)  
     $f' = \text{Augment}(f, P)$   
    Update  $f$  to  $f'$   
    Update  $G_f$  to  $G_{f'}$   
**end while**  
**return**  $f$

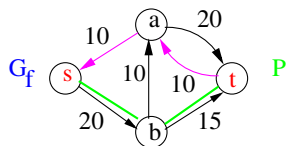
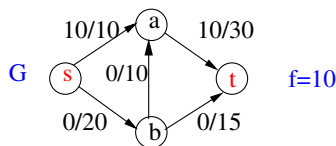


# Ford-Fulkerson algorithm

L.R. Ford, D.R. Fulkerson:  
*Maximal flow through a network*. Canadian J. of Math.  
1956.

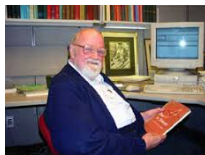


**Ford-Fulkerson**( $G, s, t, c$ )  
for all  $(u, v) \in E$  let  $f(u, v) = 0$   
 $G_f = G$   
**while** there is an  $s - t$  path in  $G_f$  **do**  
    find a simple path  $P$  in  $G_f$  (use DFS)  
     $f' = \text{Augment}(f, P)$   
    Update  $f$  to  $f'$   
    Update  $G_f$  to  $G_{f'}$   
**end while**  
**return**  $f$

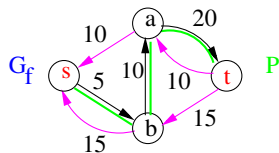
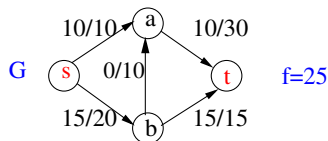


# Ford-Fulkerson algorithm

L.R. Ford, D.R. Fulkerson:  
*Maximal flow through a network*. Canadian J. of Math.  
1956.



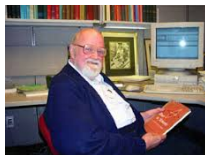
**Ford-Fulkerson**( $G, s, t, c$ )  
for all  $(u, v) \in E$  let  $f(u, v) = 0$   
 $G_f = G$   
**while** there is an  $s - t$  path in  $G_f$  **do**  
    find a simple path  $P$  in  $G_f$  (use DFS)  
     $f' = \text{Augment}(f, P)$   
    Update  $f$  to  $f'$   
    Update  $G_f$  to  $G_{f'}$   
**end while**  
**return**  $f$



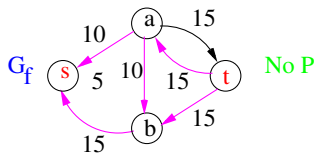
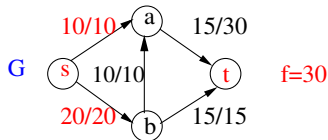


# Ford-Fulkerson algorithm

L.R. Ford, D.R. Fulkerson:  
*Maximal flow through a network*. Canadian J. of Math.  
1956.



**Ford-Fulkerson**( $G, s, t, c$ )  
for all  $(u, v) \in E$  let  $f(u, v) = 0$   
 $G_f = G$   
**while** there is an  $s - t$  path in  $G_f$  **do**  
    find a simple path  $P$  in  $G_f$  (use DFS)  
     $f' = \text{Augment}(f, P)$   
    Update  $f$  to  $f'$   
    Update  $G_f$  to  $G_{f'}$   
**end while**  
**return**  $f$



# Analysis of Ford Fulkerson

We are considering networks that initial flow and capacities are integers,

**Lemma (Integrality invariant)**

*At every iteration of the Ford-Fulkerson algorithm, the flow values  $f(e)$  and the residual capacities in  $G_f$  are integers.*

**Proof:** (induction)

- ▶ The statement is true before the **while** loop.
- ▶ Inductive Hypothesis: The statement is true after  $j$  iterations.
- ▶ iteration  $j + 1$ : As all residual capacities in  $G_f$  are integers, then bottleneck  $(P, f) \in \mathbb{Z}$ , for the augmenting path found in iteration  $j + 1$ . Thus the flow  $f'$  will have integer values  $\Rightarrow$  so will the capacities in the new residual graph.  $\square$

## Corollary: Integrality theorem

Theorem (**Integrality theorem**)

*There exists a max-flow  $f^*$  for which every flow value  $f^*$  is an integer.*

Proof:

Since the algorithm terminates, the theorem follows from the integrality invariant lemma. □

# Analysis of Ford Fulkerson

## Lemma

*If  $f$  is a flow in  $G$  and  $f'$  is the flow after an augmentation, then  $|f| < |f'|$ .*

**Proof:** Let  $P$  be the augmenting path in  $G_f$ . The first edge  $e \in P$  leaves  $s$ , and as  $G$  has no incoming edges to  $s$ ,  $e$  is a forward edge. Moreover  $P$  is simple  $\Rightarrow$  never returns to  $s$ . Therefore, the value of the flow increases in edge  $e$ .  $\square$

# Correctness of Ford-Fulkerson

Consequence of the Max-flow min-cut theorem.

## Theorem

*The flow returned by Ford-Fulkerson  $f^*$  is the max-flow.*

Proof:

- ▶ For any flow  $f$  and  $s - t$  cut  $(S, T)$  we have  $|f| \leq c(S, T)$ .
- ▶ The flow  $f^*$  is such that  $|f^*| = c(S^*, T^*)$ , for some  $s - t$  cut  $(S^*, T^*) \Rightarrow f^*$  is the max-flow. □

Therefore, for any  $(G, s, t, c)$  the value of the max  $s - t$  flow is equal to the capacity of the minimum  $s - t$  cut.

# Analysis of Ford Fulkerson: Running time

## Lemma

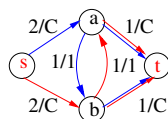
*Let  $C$  be the min cut capacity (=max. flow value), Ford-Fulkerson terminates after finding at most  $C$  augmenting paths.*

**Proof:** The value of the flow increases by  $\geq 1$  after each augmentation. □

- ▶ The number of iterations is  $\leq C$ . At each iteration:
- ▶ We have to modify  $G_f$ , with  $E(G_f) \leq 2m$ , to time  $O(m)$ .
- ▶ Using DFS, the time to find an augmenting path  $P$  is  $O(n + m)$
- ▶ Total running time is  $O(C(n + m)) = O(Cm)$
- ▶ Is that polynomial?

# Running time of Ford-Fulkerson

The number of iterations of Ford-Fulkerson could be  $\Omega(C)$   
As it is described Ford-Fulkerson can alternate  $C$  times between the blue and red paths if the figure.



$C=1000000000$

2000 million iterations  
in a G with 4 vertices!!

Recall a **pseudopolynomial** algorithm is an algorithm that is polynomial in the **unary encoding** of the input.

Is there a **polynomial time** algorithm for the max-flow problem?

# Edmonds-Karp, Dinic algorithm

J. Edmonds, R. Karp: *Theoretical improvements in algorithmic efficiency for network flow problems*. Journal ACM 1972.

Y. Dinic: *Algorithm for solution of a problem of maximum flow in a network with power estimation*. Doklady Ak.N. 1970

Choosing a **good**  
augmenting path can lead  
to a faster algorithm.

Use **BFS** to find shorter  
augmenting paths in  $G_f$ .



Using BFS on  $G_f$  we can find the shortest augmenting path  $P$  in  $O(m)$ , independently of max capacity  $C$ .



# Edmonds-Karp algorithm

Uses BFS to find the augmenting path at each  $G_f$  with fewer number of edges.

**Edmonds-Karp**( $G, s, t, c$ )

For all  $e = (u, v) \in E$  let  $f(u, v) = 0$

$G_0 = G$

**while** there is an  $s \rightsquigarrow t$  path in  $G_f$

**do**

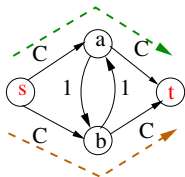
$P = \text{BFS}(G_f, s, t)$

$f' = \text{Augment}(f, P)$

Update  $G_f = G_{f'}$  and  $f = f'$

**end while**

**return**  $f$



The BFS in EK will  
choose:  $\rightarrow$  or  $\rightarrow$

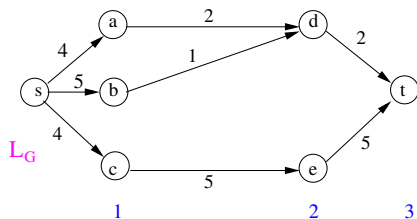
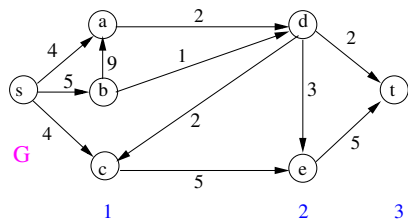
## Level graph

Given  $G = (V, E), s$ , define  $L_G = (V, E_G)$  to be its the level graph by:

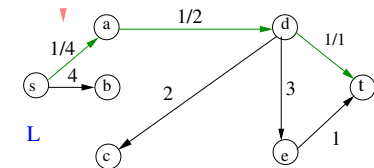
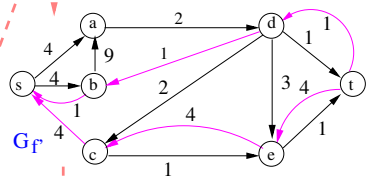
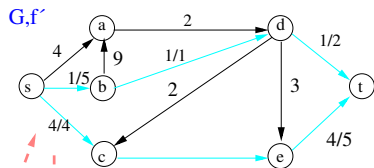
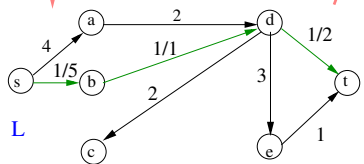
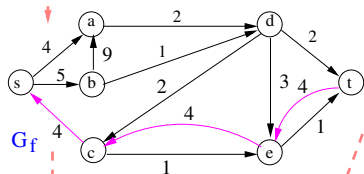
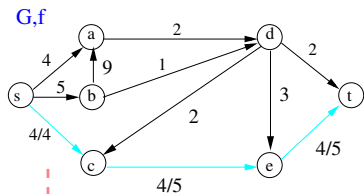
- ▶  $l(v)$  = number of edges in shortest path  $s \rightsquigarrow v$  in  $G$ ,
- ▶  $L_G = (V, E_G)$  is the subgraph of  $G$  that contains only edges  $(v, w) \in E$  s.t.  $l(w) = l(v) + 1$ .

Notice:

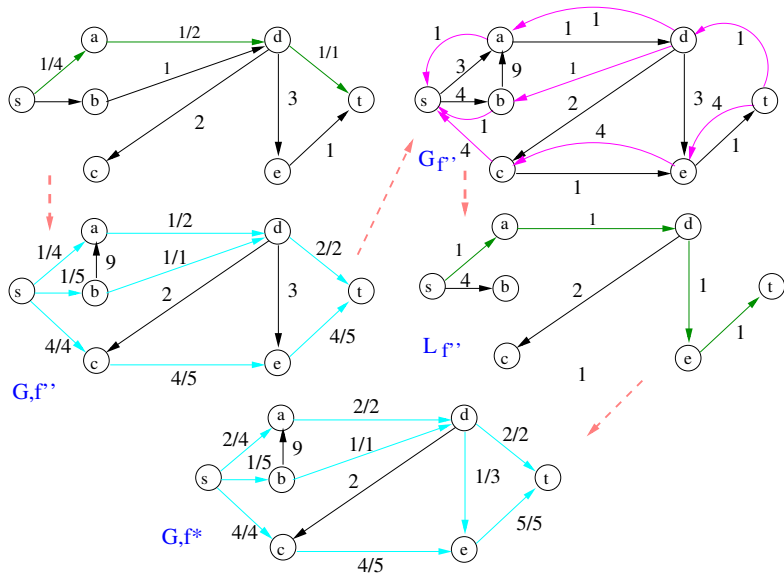
- ▶ Using BFS we can compute  $L_G$  in  $O(n + m)$
- ▶ **Important property:**  $P$  is a shortest  $s \rightsquigarrow t$  in  $G$  iff  $P$  is an  $s \rightsquigarrow t$  path in  $L_G$ .



# The working of the EK algorithm



# The working of the EK algorithm



# EK algorithm: Properties

## Lemma

*Throughout the algorithm, the length of the shortest path never decreases.*

### Proof:

- ▶ Let  $f$  and  $f'$  be the flow before and after a shortest path augmentation
- ▶ let  $L$  and  $L'$  be the levels graphs of  $G_f$  and  $G_{f'}$ .
- ▶ Only back edges added to  $G_{f'}$ . □

## Lemma

*After at most  $m$  shortest path augmentations, the length of  $P$  is monotonically increasing.*

### Proof:

- ▶ The bottleneck edge is deleted from  $L$  after each augmentation.
- ▶ No new edge is added to  $L$  until length of shortest path strictly increases

# Complexity of Edmonds-Karp algorithm

Using the the previous lemmas, we prove

## Theorem

*The EK algorithms runs in  $O(m^2n)$  steps. Therefore it is a polynomial time algorithm.*

Proof:

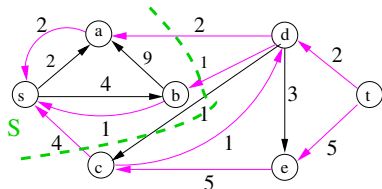
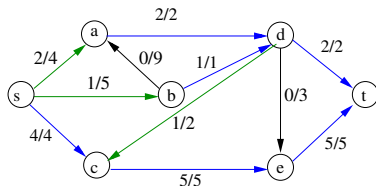
- ▶ Need time  $O(m + n)$  to find the augmenting path using BFS.
- ▶ Need  $O(m)$  augmentations for paths of length  $k$ .
- ▶ Every augmentation path is simple  $\Rightarrow 1 \leq k \leq n \Rightarrow O(nm)$  augmentations □

## Finding a min-cut

Given  $(G, s, t, c)$  to find a min-cut:

1. Compute the max-flow  $f^*$  in  $G$ .
2. Obtain  $G_{f^*}$ .
3. Find the set  $S = \{v \in V \mid s \rightsquigarrow v\}$  in  $G_{f^*}$ .
4. Output the cut  
 $(S, V - \{S\}) = \{(v, u) \mid v \in S \text{ and } u \in V - \{S\}\}$  in  $G$ .

The running time is the same than the algorithm to find the max-flow.



# The max-flow problems: History

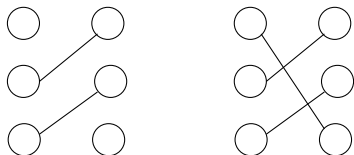
- ▶ Ford-Fulkerson (1956)  $O(mC)$ , where  $C$  is max capacity.
- ▶ Dinic (1970) (blocking flow)  $O(n^2m)$
- ▶ Edmond-Karp (1972) (shortest augmenting path)  $O(nm^2)$
- ▶ Karzanov (1974),  $O(n^2m)$  Goldberg-Tarjan (1986) (push re-label preflow + dynamic trees)  $O(nm \lg(n^2/m))$  (for this time it uses parallel implementation)
- ▶ King-Rao-Tarjan (1998)  $O(nm \log_{m/n} n)$ .
- ▶ J. Orlin (2013)  $O(nm)$  (clever follow up to KRT-98)



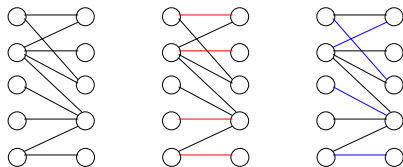


## Maximum matching in graphs bipartite

A graph  $G = (V, E)$  is said to be **bipartite** if  $V$  can be partite in  $L$  and  $R$ ,  $L \cup R = V$ ,  $L \cap R = \emptyset$ , such that every  $e \in E$  connects  $L$  with  $R$ .



The **max matching bipartite graph problem**: given a bipartite  $G = (L \cup R, E)$  with  $2n$  vertices find a maximum matching.

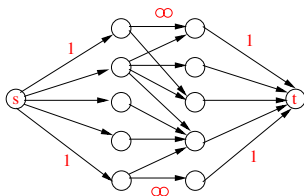


Max matchings = 4

## Maximum matching: flow formulation

Given a bipartite graph  $G = (L \cup R, E)$  construct  $\hat{G} = (\hat{V}, \hat{E})$ :

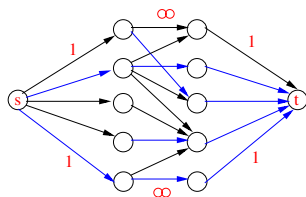
- ▶ Add vertices  $s$  and  $t$ :  $\hat{V} = L \cup R \cup \{s, t\}$ .
- ▶ Add directed edges  $s \rightarrow L$  with capacity 1. Add directed edges  $R \rightarrow t$  with capacity 1.
- ▶ Direct the edges  $E$  from  $L$  to  $R$ , and give them capacity  $\infty$ .
- ▶  $\hat{E} = \{s \rightarrow L\} \cup E \cup \{R \rightarrow t\}$ .



## Maximum matching: flow formulation

Given a bipartite graph  $G = (L \cup R, E)$  construct  $\hat{G} = (\hat{V}, \hat{E})$ :

- ▶ Add vertices  $s$  and  $t$ :  $\hat{V} = L \cup R \cup \{s, t\}$ .
- ▶ Add directed edges  $s \rightarrow L$  with capacity 1. Add directed edges  $R \rightarrow t$  with capacity 1.
- ▶ Direct the edges  $E$  from  $L$  to  $R$ , and give them capacity  $\infty$ .
- ▶  $\hat{E} = \{s \rightarrow L\} \cup E \cup \{R \rightarrow t\}$ .

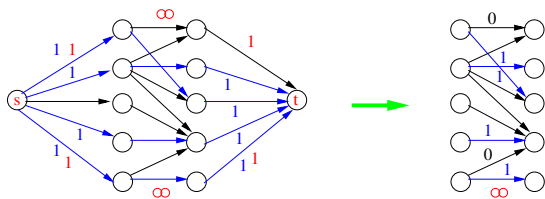




# Maximum matching: Analysis

## Max flow $\leq$ Max bipartite matching

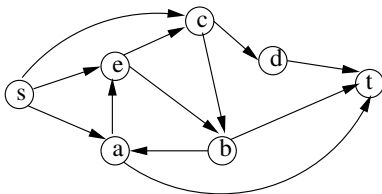
- ▶ If there is a flow  $f$  in  $\hat{G}$ ,  $|f| = k$ , as capacities are  $\mathbb{Z}^*$   $\Rightarrow$  an integral flow exists.
- ▶ Consider the cut  $C = (\{s\} \cup L, R \cup \{t\})$  in  $\hat{G}$ .
- ▶ Let  $F$  be the set of edges in  $C$  with flow=1, then  $|F| = k$ .
- ▶ Each node in  $L$  is in at most one  $e \in F$  and every node in  $R$  is in at most one head of an  $e \in F$
- ▶ Therefore, exists a bipartite matching  $F$  in  $G$  with  $|F| \leq |f|$   $\square$



# Disjoint path problem

Given a digraph  $(G = (V, E), s, t)$ , a set of paths is **edge-disjoint** if their edges are disjoint (although they may go through some of the same vertices)

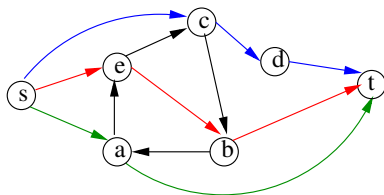
The disjoint path problem given  $G, s, t$  find the **max number of edge disjoint paths**  $s \rightsquigarrow t$



# Disjoint path problem

Given a digraph  $(G = (V, E), s, t)$ , a set of paths is **edge-disjoint** if their edges are disjoint (although they may go through some of the same vertices)

The disjoint path problem given  $G, s, t$  find the **max number of edge disjoint paths**  $s \rightsquigarrow t$



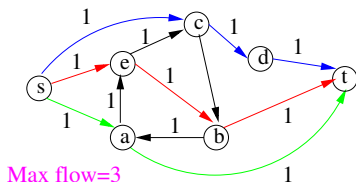
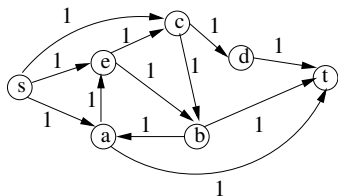


# Disjoint path problem: Max flow formulation

Assign unit capacity to every edge

Theorem

*The max number of edge disjoint paths  $s \rightsquigarrow t$  is equal to the max flow value*



# Disjoint path problem: Proof of the Theorem

Number of disjoint paths  $\leq$  max flow

If we have  $k$  edge-disjoint paths  $s \rightsquigarrow t$  in  $G$  then making  $f(e) = 1$  for each  $e$  in a path, we get a flow  $= k$

Number of disjoint paths  $\geq$  max flow

If max flow  $|f^*| = k \Rightarrow \exists$  0-1 flow  $f^*$  with value  $k$

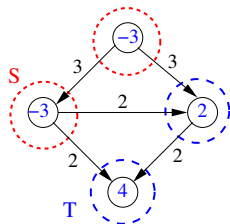
$\Rightarrow \exists k$  edges  $(s, v)$  s.t.  $f(s, v) = 1$ , by **flow conservation** we can extend to  $k$  paths  $s \rightsquigarrow t$ , where each edge is a path carries flow  $= 1$ . □

If we have an undirected graph, with two distinguished nodes  $u, v$ , how would you apply the max flow formulation to solve the problem of finding the max number of disjoint paths between  $u$  and  $t$ ?

# Circulation with demands

Given a graph  $G = (V, E)$  with capacities  $c$  in the edges, such that each  $v \in V$  is associate with a demand  $d(v)$ , where

- ▶ If  $d(v) > 0 \Rightarrow v$  is a **sink**,  $v$  can receive  $d(v)$  units of flow more than it sends.
- ▶ If  $d(v) < 0 \Rightarrow v$  is a **source**,  $v$  can send  $d(v)$  units of flow more than it receives.
- ▶ If  $d(v) = 0$  then  $v$  is neither a source or a sink.
- ▶ Define  $S$  to be the set of sources and  $T$  the set of sinks.

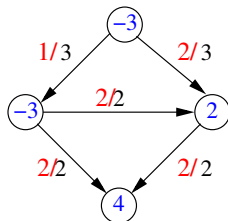


## Circulation with demands problem

Given  $G = (V, E)$  with  $c \geq 0$  and  $\{d(v)\}_{v \in V}$ , define a **circulation** as a function  $f : E \rightarrow \mathbb{R}^+$  s.t.

1. **capacity:** For each  $e \in E$ ,  $0 \leq f(e) \leq c(e)$ ,
2. **conservation:** For each  $v \in V$ ,

$$\sum_{(u,v) \in E} f(u,v) - \sum_{(v,z) \in E} f(v,z) = d(v).$$



**Circulation with demands feasibility problem:** Given  $G = (V, E)$  with  $c \geq 0$  and  $\{d(v)\}_{v \in V}$ , does it exist a feasible circulation?

**Feasible circulation:** a function  $f$  on  $G$  with  $c \geq 0$  and  $\{d(v)\}_{v \in V}$ , such that it satisfies (1) and (2)?

## Circulation with demands problem

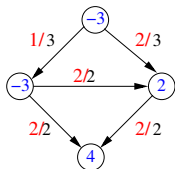
Notice that if  $f$  is a feasible circulation, then

$$\sum_{v \in V} d(v) = \sum_{v \in V} \left( \underbrace{\sum_{(u,v) \in E} f(u,v)}_{\text{edges to } v} - \underbrace{\sum_{(v,z) \in E} f(v,z)}_{\text{edges out of } v} \right).$$

Notice  $\sum_{v \in V} d(v) = 0$ , so we have,

So If there is a feasible circulation with demands  $\{d(v)\}_{v \in V}$ , then  $\sum_{v \in V} d(v) = 0$ .

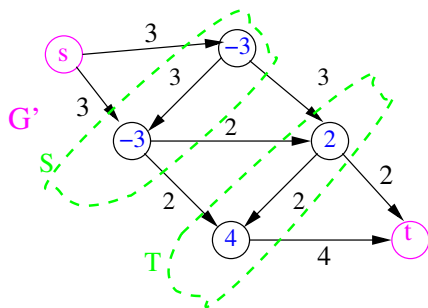
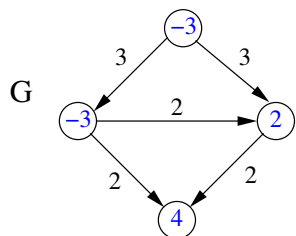
Therefore as  $S = \{v \in V \mid d(v) > 0\}$  and  $T = \{v \in V \mid d(v) < 0\}$ , we can define  $D = -\sum_{v \in S} d(v) = \sum_{v \in T} d(v)$ .



## Circulation with demands: Max-flow formulation

Extend  $G = (V, E)$  to  $G' = (V', E')$  by

- ▶ Add new source  $s$  and sink  $t$ .
- ▶ For each  $v \in S$  ( $d(v) < 0$ ) add  $(s, v)$  with capacity  $-d(v)$ .
- ▶ For each  $v \in T$  ( $d(v) > 0$ ) add  $(v, t)$  with capacity  $d(v)$ .



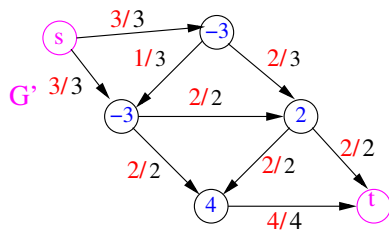
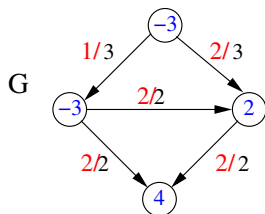
# Analysis

1.- Every flow  $f : s \rightsquigarrow t$  in  $G'$  must be  $|f| \leq D$

The capacity  $c(\{s\}, V) = D \Rightarrow$  by max-flow min-cut Thm. any max-flow  $f$  in  $G'$ ,  $|f| \leq D$ .

2.- If there is a feasible circulation  $f$  with  $\{d(v)\}_{v \in V}$  in  $G$ , then we have a max-flow  $f : s \rightsquigarrow t$  in  $G'$  with  $|f| = D$

$\forall (s, v) \in E', f'(s, v) = -d(v)$  and  $\forall (u, t) \in E', f'(u, t) = d(v)$ .

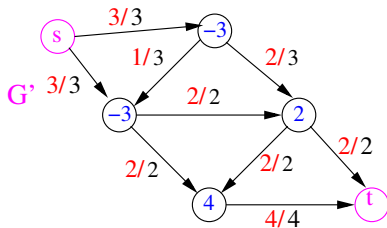
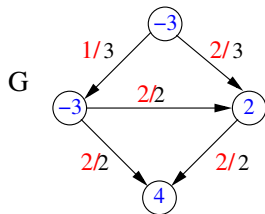


# Analysis

3.- If there is a flow  $f' : s \rightsquigarrow t$  in  $G'$  with  $|f'| = D$ :

1.  $\forall (s, v) \in E'$  and  $\forall (u, t) \in E'$  must be saturated  $\Rightarrow$  if we delete these edges in  $G'$  we obtain a circulation  $f$  in  $G$ .

2.  $f$  satisfies  $d(v) = \underbrace{\sum_{(u,v) \in E} f(u,v)}_{\text{edges to } v} - \underbrace{\sum_{(v,z) \in E} f(v,z)}_{\text{edges out of } v}$ .





# Main results

## Theorem (Circulation integrality theorem)

*If all capacities and demands are integers, and there exists a circulation, then there exists an integer valued circulation.*

Sketch Proof Max-flow formulation + integrality theorem for max-flow □

From the previous discussion, we can conclude:

## Theorem (Necessary and sufficient condition)

*There is a feasible circulation with  $\{d(v)\}_{v \in V}$  in  $G$  iff the max-flow in  $G'$  has value  $D$ .*

# Circulations with demands and lower bounds: Max-flow formulation

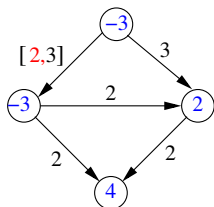
Generalization of the previous problem: besides satisfy demands at nodes, we want to force the flow to use certain edges.

Introduce a new constrain  $\ell(e)$  on each  $e \in E$ , indicating the min-value the flow must be on  $e$ .

Given  $G = (V, E)$  with  $c(e)$ ,  $c(e) \geq \ell(e) \geq 0$ , for each  $e \in E$  and  $\{d(v)\}_{v \in V}$ , define a **circulation** as a function  $f : E \rightarrow \mathbb{R}^+$  s.t.

1. **capacity**: For each  $e \in E$ ,  
 $\ell(e) \leq f(e) \leq c(e)$ ,
2. **conservation**: For each  $v \in V$ ,

$$\sum_{(u,v) \in E} f(u,v) - \sum_{(v,z) \in E} f(v,z) = d(v).$$

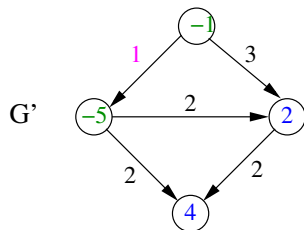
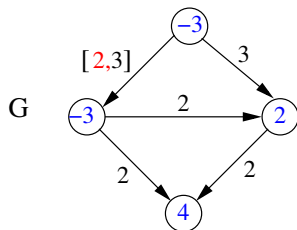


Circulation problems with lower bounds: Given  $(G, c, \ell, \{d(v)\})$ , does there exist a feasible circulation?

# Circulations with demands and lower bounds: Max-flow formulation

Let  $(G = (V, E), c, \ell, d(\cdot))$  be a graph, construct  $G' = (V, E), c', d'$ , where for each  $e = (u, v) \in E$ , with  $\ell(e) > 0$ :

- ▶  $c'(e) = c(e) - \ell(e)$  (sent  $\ell(e)$  units along  $e$ ).
- ▶ Update the demands on both ends of  $e$  ( $d'(u) = d(u) + \ell(e)$  and  $d'(v) = d(v) - \ell(e)$ )



# Main result

## Theorem

*There exists a circulation in  $G$  iff there exists a circulation in  $G'$ .  
Moreover, if all demands, capacities and lower bounds in  $G$  are integers, then there is a circulation in  $G$  that is integer-valued.*

**Sketch Proof** Need to prove  $f(e)$  is a circulation in  $G$  iff  
 $f'(e) = f(e) - \ell(e)$  is a circulation in  $G'$ .

The integer-valued circulation part is a consequence of the integer-value circulation Theorem for  $f'$  in  $G'$ . □

# Survey design problem

Problem: Design a survey among customers of products

- ▶ Each customer will receive questions about some products.
- ▶ Each customer  $i$  can only be asked about a number of products between  $c_i$  and  $c'_i$  ( $[c_i, c'_i]$ ) which he has purchased.
- ▶ For each product  $j$  we want to collect data for a minimum of  $p_j$  distinct customers and a maximum of  $p'_j$  ( $[p_j, p'_j]$ )



# Survey design problem

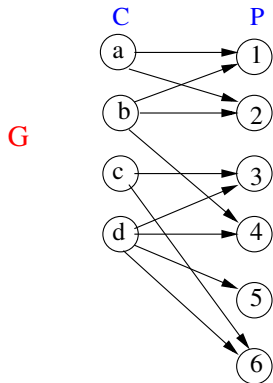
Measuring customer satisfaction.

Consider  $n$  customers and  $m$  products.

Formally we want to model the problem as:

- ▶ A bipartite graph  $G = (C \cup P, E)$ , where  $C = \{i\}$  is the set of customers and  $P = \{j\}$  is the set of products.
- ▶ There is an  $(i, j) \in E$  if  $i$  has purchased product  $j$ .
- ▶ For each  $i \in \{1, \dots, n\}$ , we have bounds  $[c_i, c'_i]$  on the number of products  $i$  can be asked about.
- ▶ For each  $j \in \{1, \dots, m\}$ , we have bounds  $[p_j, p'_j]$  on the number of customers that can be asked about it.

# Survey design problem: Bipartite graph $G$



Customers  $C = \{a, b, c, d\}$

Products  $P = \{1, 2, 3, 4, 5, 6\}$

Customer	Buys
a	1,2
b	1,2,4
c	3,6
d	3,4,5,6

a:[1,2]

1: [1,2]

b:[1,3]

2: [1,2]

c:[1,2]

3: [1,2]

d:[2,4]

4: [1,2]

5: [0,1]

6: [1,2]

# Survey design problem: Max flow formulation

We construct  $G'$  from  $G$ , by adding:

Edges:  $s \rightarrow \{C\}$ ,  $\{P\} \rightarrow t$ , and

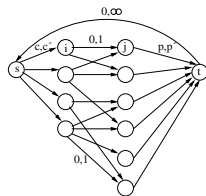
$(t, s)$ .

Capacities:  $c(t, s) = \infty$

$c(i, j) = 1$ ,

$c(s, i) = [c_i, c'_i]$ ,

$c(j, t) = [p_j, p'_j]$ .

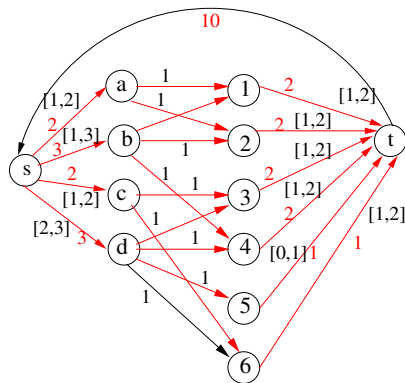


Notice if  $f$  is the flow:

- ▶  $f(i, j) = 1 \Rightarrow$  customer  $i$  is asked about product  $j$ ,
- ▶  $f(s, i) = \#$  products to ask customer  $i$  for opinion,
- ▶  $f(j, t) = \#$  customers to be asked to review product  $j$ ,
- ▶  $f(t, s)$  is the number of questions asked.



# Max flow formulation: Example



a → 1,2

a:[1,2]

1: [1,2]

b → 1,2,4

b:[1,3]

2: [1,2]

c → 3,6

c:[1,2]

3: [1,2]

d → 3,4,5,6

d:[2,3]

4: [1,2]

5: [0,1]

6: [1,2]

# Main result

**Theorem**  $G'$  has a feasible circulation iff there is a feasible way to design the survey.

**Proof** if there is a feasible way to design the survey:

- ▶ if  $i$  is asked about  $j$  then  $f(i, j) = 1$ ,
- ▶  $f(s, i) =$  number questions asked to  $i$ ,
- ▶  $f(j, t) =$  number of customers who were asked about  $j$ ,
- ▶  $f(t, s) =$  total number of questions.
- ▶ easy to verify that  $f$  is feasible in  $G'$

If there is an integral, feasible circulation in  $G'$ :

- ▶ if  $f(i, j) = 1$  then  $i$  will be asked about  $j$ ,
- ▶ the constrains  $(c_i, c'_i, p_j, p'_j)$  will be satisfied. □

# Conclusions

Max-Flow/ Min-Cut problem is an intuitively easy problem with lots of applications.

We just presented a few ones.

An alternative point of view can be obtained from duality in Linear Programming

The material in this talk has been basically obtained from two textbooks:

- ▶ Chapter 26 of Cormen, Leiserson, Rivest, Stein: [Introduction to Algorithms](#), and
- ▶ Chapter 7 of Kleinberg, Tardos: [Algorithm Design](#).