Analysis of Algorithms

V. Adamchik

Lecture 9

Spring 2023

CSCI 570

University of Southern California

# Network Flow - 2

Reading: chapter 7

# The Ford-Fulkerson Algorithm

Algorithm. Given (G, s, t, c)

start with f(u,v)=0 and $G_f$ = G.

*while* exists an augmenting s-t path in $G_f$

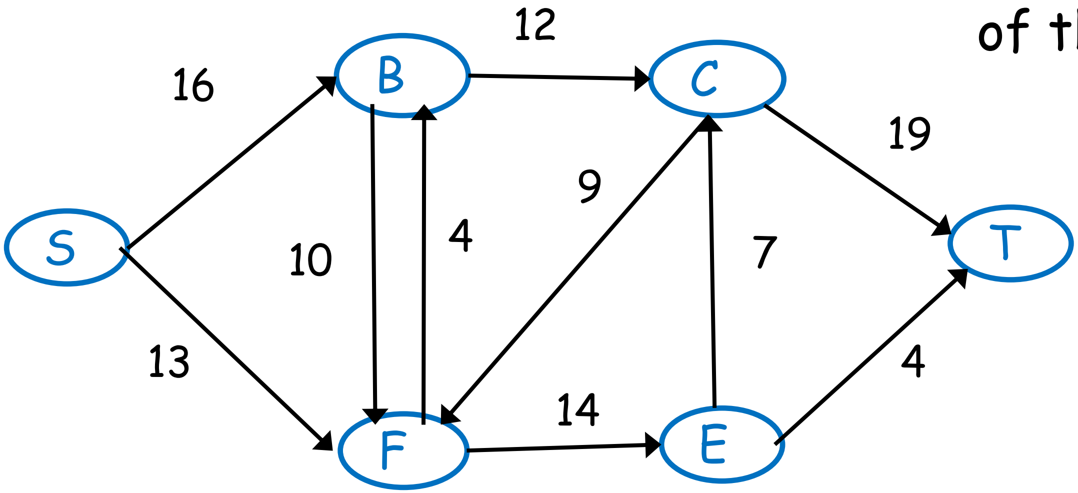
find a bottleneck

augment the flow along this path

update the residual graph $G_f$

$O(|f|\cdot(E+V))$

It is pseudo-polynomial because it depends on the size of the integers |f| in the input.




$$|f| = \sum_{e \text{ out of } s} f(e)$$

# Duality

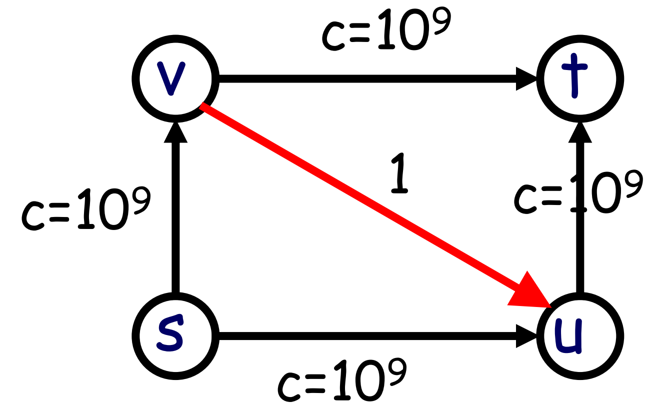
max-flow min-cut

Lemma 2: $|f| \leq cap(A, B)$

Theorem FF: $|f| = cap(A, B)$

# How to improve the efficiency?

In the FF algorithm we run DFS.
What about if we run BFS?
BFS will return the shortest path in the number of edges.




This variation is called the Edmonds-Karp algorithm

It can be shown that this requires only O(V E) iterations. The proof is beyond the scope of 570.

The total runtime: $O(V \cdot E^2)$, it's polynomial!!!

# Edmonds-Karp algorithm

Algorithm. Given $(G, s, t, c)$

1) Start with $|f|=0$, so $f(e)=0$
2) Find a shortest augmenting path in $G_f$
3) Augment flow along this path
4) Repeat until there is no an s-t path in $G_f$

Theorem.

The runtime complexity of the algorithm is $O(V E^2)$. $= O(V^5)$

(without proof)

# Runtime history

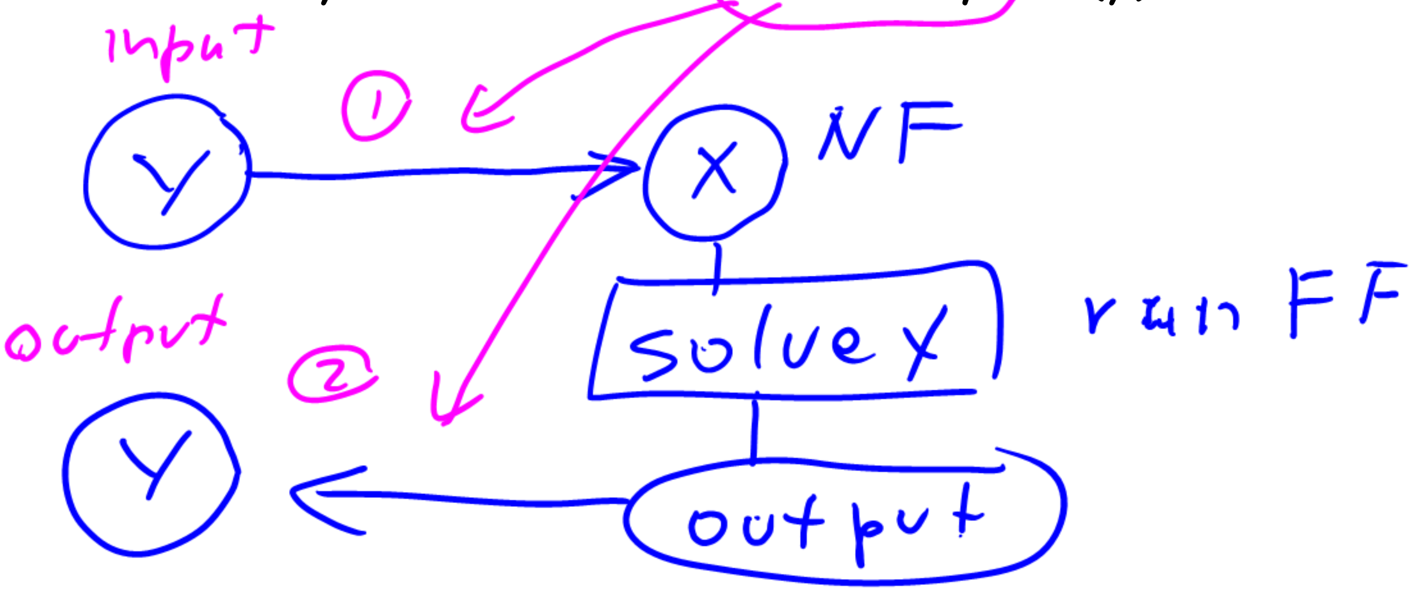
n = V, m = E,
U = |f|

| year | discoverer(s) | bound |
|---|---|---|
| 1951 | Dantzig [11] | $O(n^2 mU)$ |
| 1956 | $\boxed{\text{Ford \& Fulkerson [17]}}$ | O(m U) |
| 1970 | Dinitz [13]<br>Edmonds & Karp [15] | O(n m²) shortest path |
| 1970 | Dinitz [13] | $O(n^2 m)$ |
| 1972 | Edmonds & Karp [15]<br>Dinitz [14] | $O(m^2 \log U)$ capacity scaling |
| 1973 | Dinitz [14]<br>Gabow [19] | $O(nm \log U)$ |
| 1974 | Karzanov [36] | $O(n^3)$ preflow-push |
| 1977 | Cherkassky [9] | $O(n^2 m^{1/2})$ |
| 1980 | Galil & Naamad [20] | $O(nm \log^2 n)$ |
| 1983 | Sleator & Tarjan [46] | $O(nm \log n)$ splay tree |
| 1986 | Goldberg & Tarjan [26] | $O(nm \log(n^2/m))$ preflow-push API |
| 1987 | Ahuja & Orlin [2] | $O(nm + n^2 \log U)$ |
| 1987 | Ahuja et al. [3] | $O(nm \log(n\sqrt{\log U}/m))$ |
| 1989 | Cheriyan & Hagerup [7] | $E(nm + n^2 \log^2 n)$ |
| 1990 | Cheriyan et al. [8] | $O(n^3/\log n)$ |
| 1990 | Alon [4] | $O(nm + n^{8/3} \log n)$ |
| 1992 | King et al. [37] | $O(nm + n^{2+\epsilon})$ |
| 1993 | Phillips & Westbrook [44] | $O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$ |
| 1994 | King et al. [38] | $O(nm \log_{m/(n \log n)} n)$ |
| 1997 | Goldberg & Rao [24] | $O(\min(n^{2/3}, m^{1/2}) m \log(n^2/m) \log U)$ |
| 2013 | Orlin | O(m n) |

# Reduction

Formally, to reduce a problem $Y$ to a problem $X$ (we write $Y \leq_p X$) we want a function $f$ that maps $Y$ to $X$ such that:

- $f$ is a <u>polynomial</u> time computable
- $\forall$ *instance* $y \in Y$ is solvable if and only if $f(y) \in X$ is solvable.



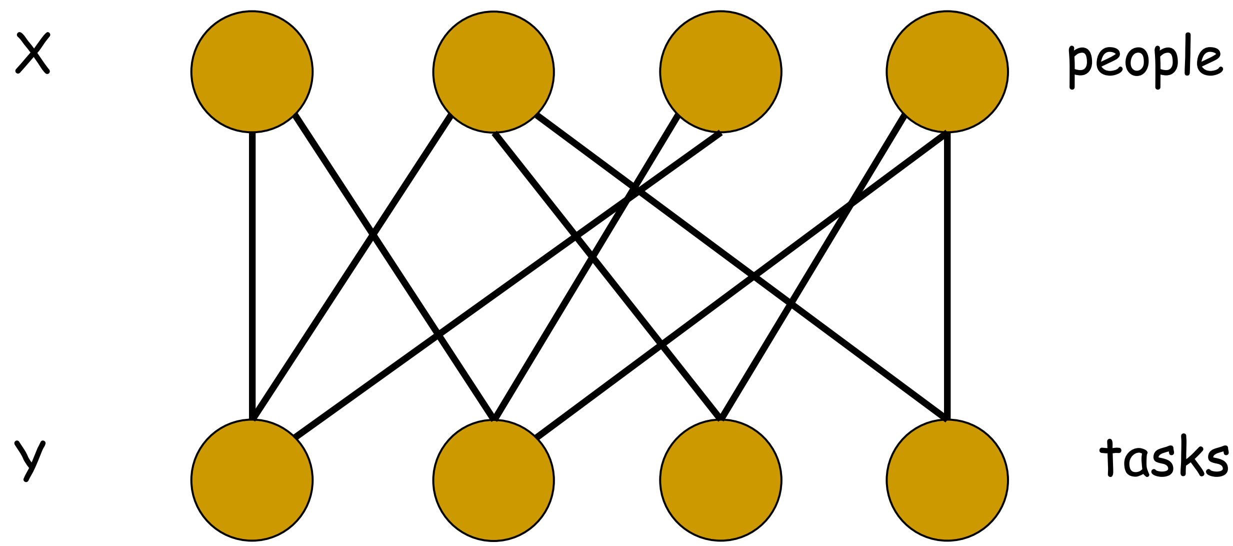
# Solving by reduction to NF

1. Describe how to construct a flow network.

2. Make a claim. Something like "this problem has a feasible solution if and only if the max flow is ...".

3. Prove the above claim in <u>both</u> directions.

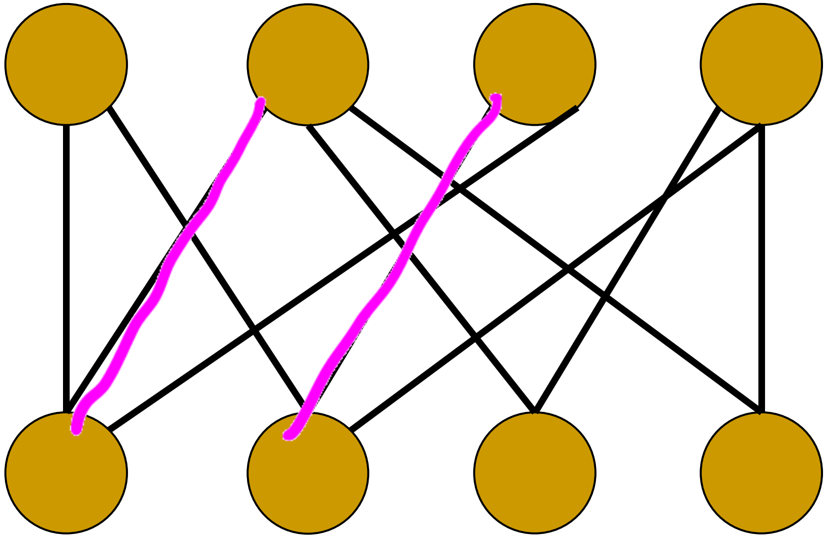
# Bipartite Graph




A graph is bipartite if the vertices can be partitioned into two disjoint (also called independent) sets X and Y such that all edges go only between X and Y (no edges go from X to X or from Y to Y). Often, we write G = (X, Y, E).

# Bipartite Matching

Definition. A subset of edges is a matching if no two edges have a common vertex (mutually disjoint).



Definition. A maximum matching is a matching with the largest possible number of edges.

Goal. Find a maximum matching in G.

We will solve this problem by reduction.
Given an instance of bipartite matching, we will create an instance of network flow. The solution to that network flow problem will be used to find the solution to the bipartite matching problem.

# Reducing Bipartite Matching to Network Flow

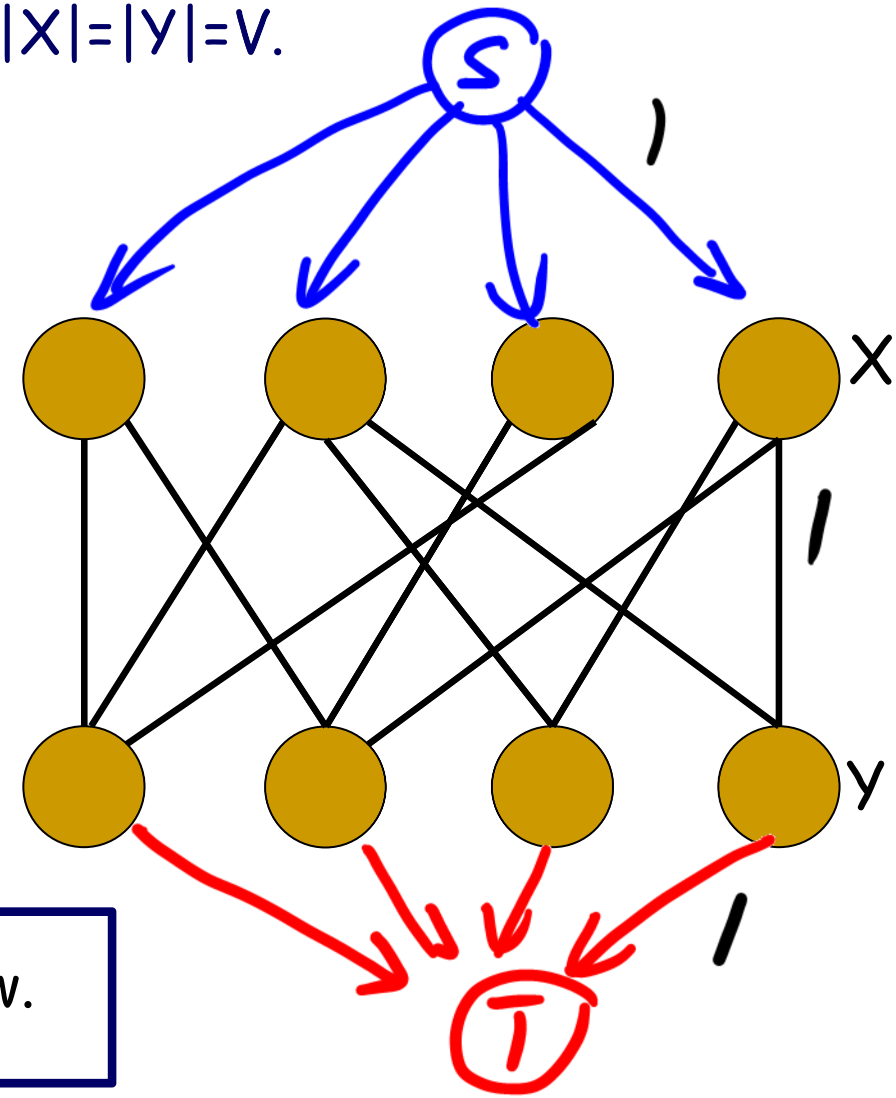
Given bipartite $G = (X, Y, E)$. Let $|X|=|Y|=V$.

$\forall e \in E$, direct edges from X to Y.

Create a new vertex S with outgoing directed edges.

Create a new vertex T with incoming directed edges.

Let each edge has capacity equal to 1.




Claim: Max matching = Max flow.

# (if then) Max matching $\Rightarrow$ Max flow

*If there is a matching of k edges, then there is a flow of value k.*

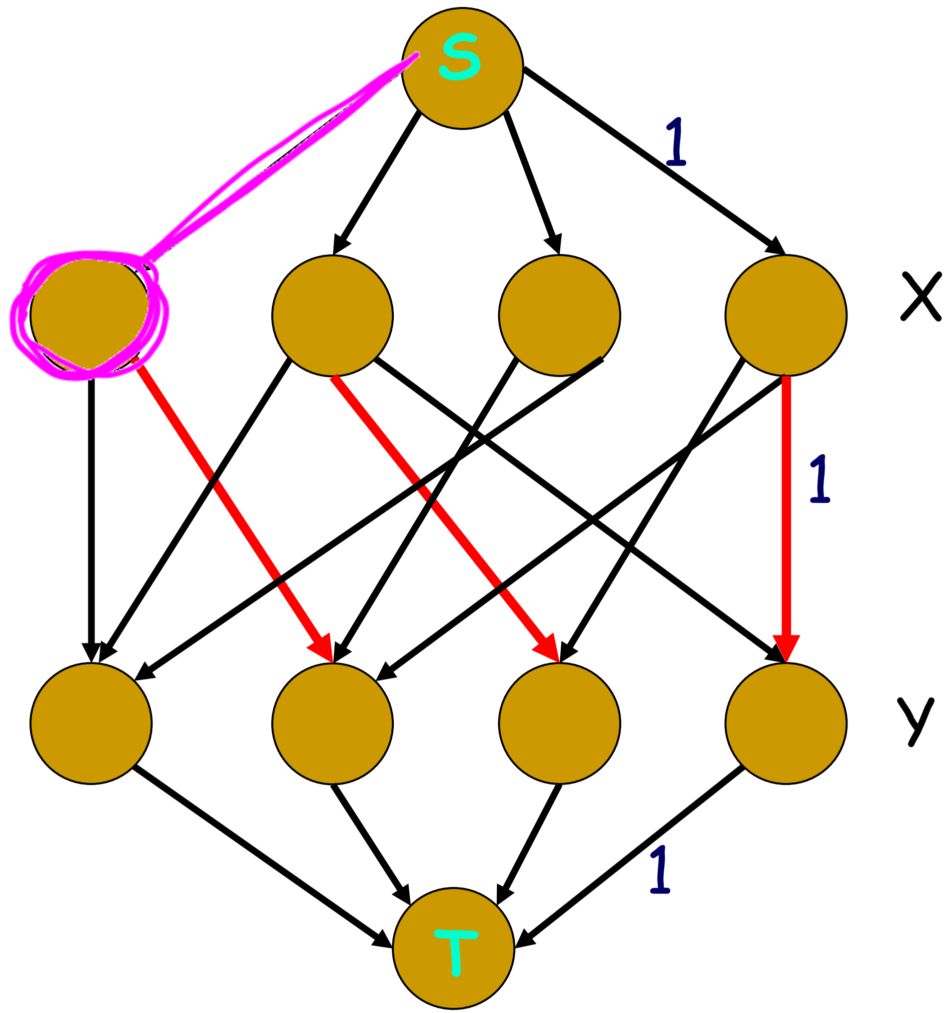
Proof.

Push a flow (in red) over matching.

f has 1 unit of flow across each edge.

Either 0 or 1 unit leaves & enters each node (except s, t).

By conservation constraint, it follows that we have a flow of value k.



# Max matching $\Leftarrow$ Max flow

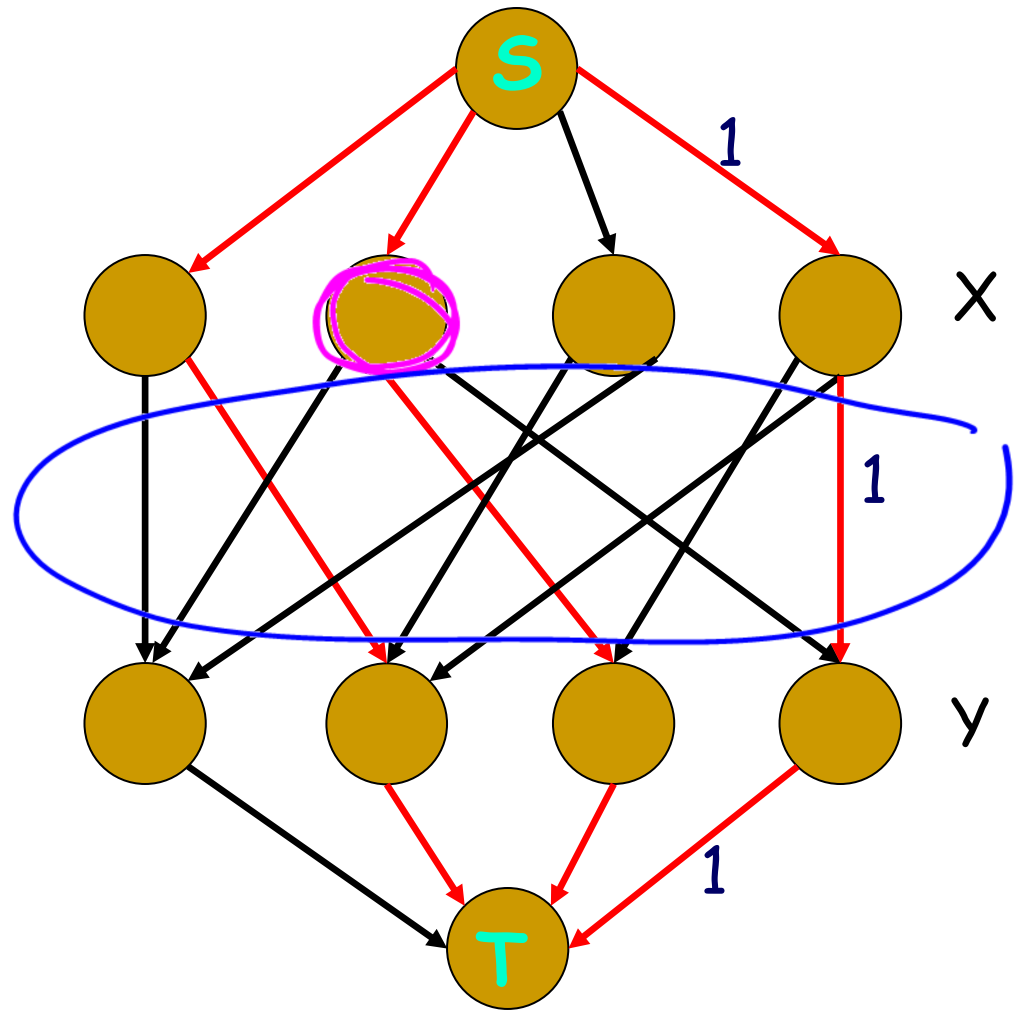
*If there is a flow f of value k, there is a matching with k edges.*

Proof.

Recall Lemma 2.

For any flow and cut

$$|f| = \sum_{e \text{ out of } X} f(e) - \sum_{e \text{ in to } X} f(e)$$



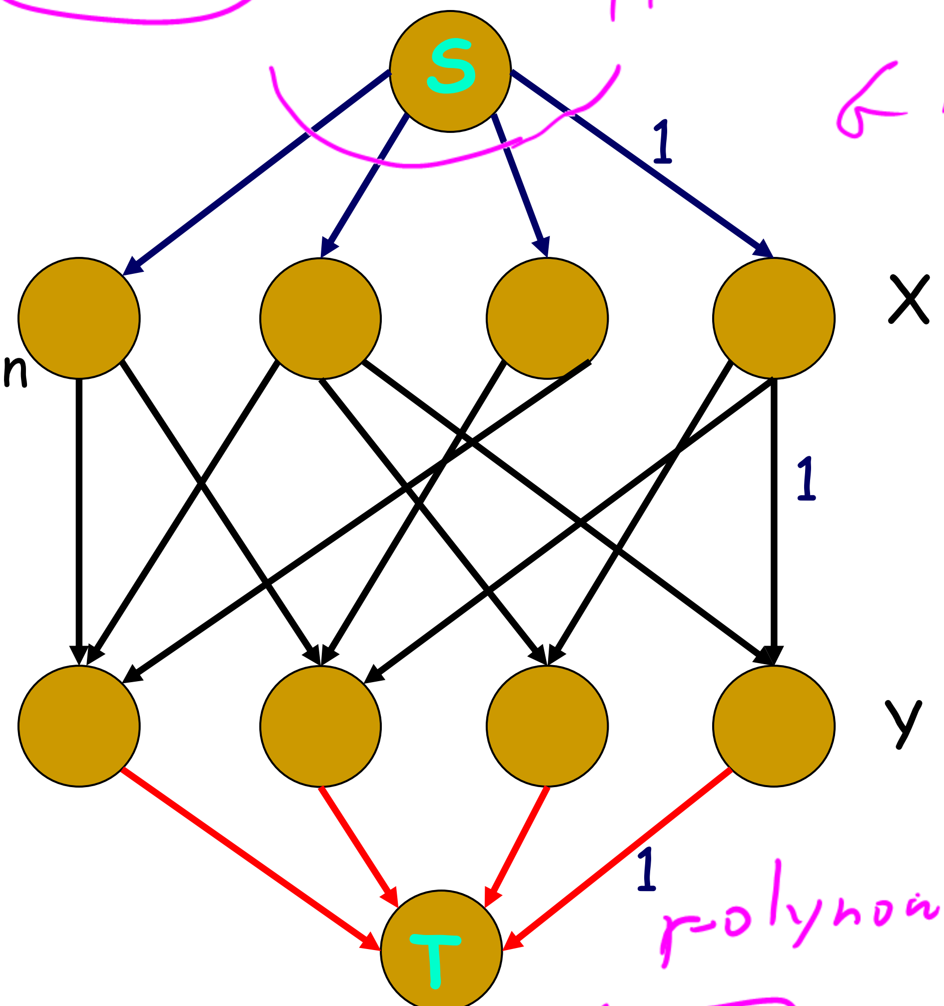
# Runtime Complexity

Given bipartite $G = (X, Y, E)$. Let $|X|=|Y|=V$.

$|f|$

How long does it take to solve the network flow problem on the new graph $G' = (V', E')$ (on the right)?

$G'$




The running time of Ford-Fulkerson is

$O(|f|\ (E' + V'))$ on $G'$

where

$|f| = V$, size of X.

$V' = 2V + 2$

$E' = E + 2V$.

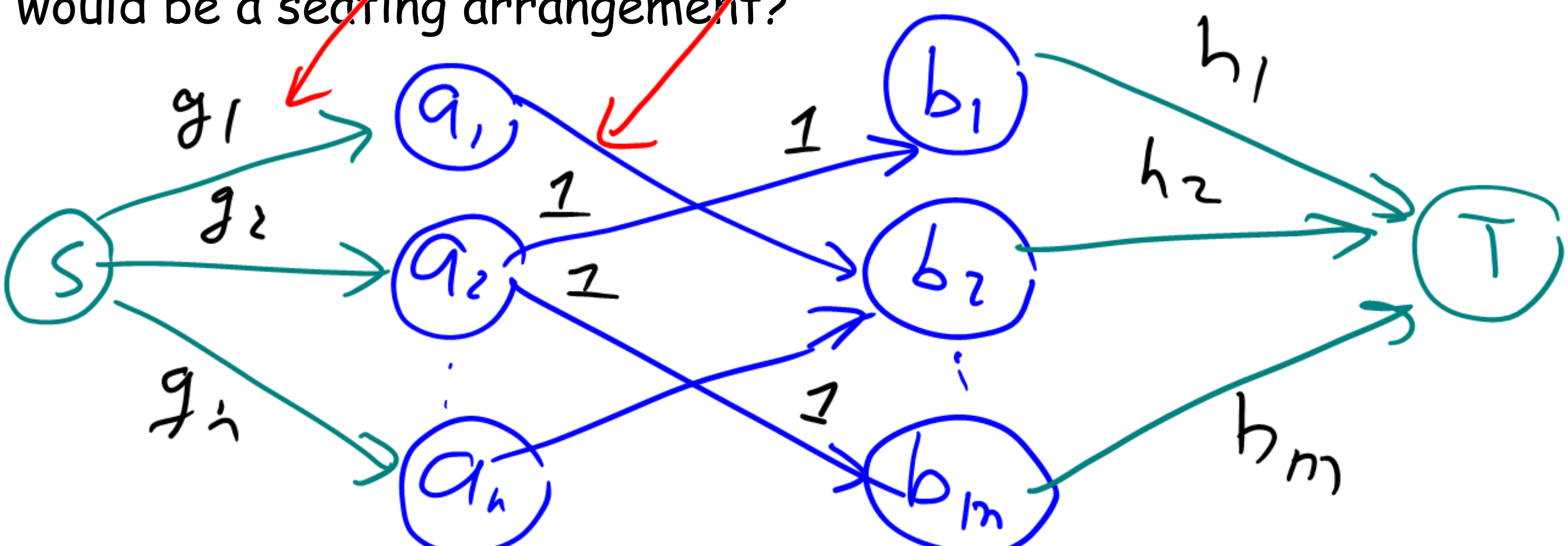
polynomial

So, the runtime is $O(V\ (E + 2V + 2V + 2)) = O(V\ E + V^2) = \boxed{O(V\ E)}$

# Discussion Problem 1 Break

At a dinner party, there are $n$ families $a_1, a_2, \ldots, a_n$ and $m$ tables $b_1, b_2, \ldots, b_m$. The $i$-th family $a_i$ has $g_i$ members and the $j$-th table $b_j$ has $h_j$ seats. Everyone is interested in making new friends and the dinner party planner wants to seat people such that no two members of the same family are seated at the same table. Design an algorithm that decides if there exists a seating assignment such that everyone is seated and no two members of the same family are seated at the same table. What would be a seating arrangement?

1



② 

Claim. An assignment $\exists$

if and only if

the max-flow $= g_1 + g_2 + \dots + g_n$

③

Proof.

$\Rightarrow$) Given: an assignment

Prove: the max-flow

$\exists$ assignment $\rightarrow$ everyone is seated

$\rightarrow$ edges from the source are saturated

$\rightarrow$ max-flow $= g_1 + g_2 + \dots + g_n$

$\Leftarrow$) Given: max-flow

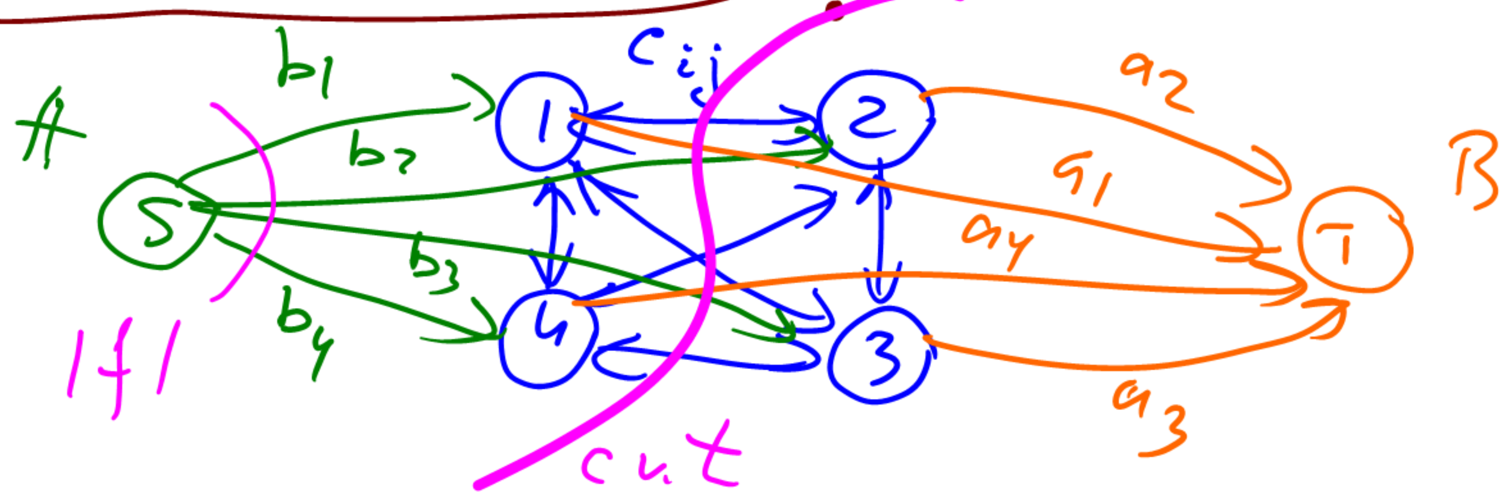
Prove: assignment

# Discussion Problem 2

$A$ company has $n$ locations in city $A$ and plans to move some of them (or all) to another city $B$. The $i$-th location costs $a_i$ per year if it is in the city $A$ and $b_i$ per year if it is in the city $B$. The company also needs to pay an extra cost, $c_{ij} > 0$, per year for traveling between locations $i$ and $j$. We assume that $c_{ij} = c_{ji}$. Design an efficient algorithm to decide which company locations in city $A$ should be moved to city $B$ in order to minimize the total annual cost.

?

1



$cap(A,B) = b_2 + b_3 + a_1 + a_4 + c_{12} + c_{24} + c_{34} + c_{13}$

Goal: $\min cap(A,B)$

2

**Claim.** The total moving cost is the min if and only if

$$\text{max.flow} = \underbrace{\sum_{j \in B} b_j}_{b_2+b_3} + \underbrace{\sum_{i \in A} a_i}_{a_1+a_4} + \sum_{\substack{i \in A \\ j \in B}} c_{ij}$$

**Proof.**

$\Rightarrow$

$\Leftarrow$ Proved in the FF theorem

max-flow = min-cut = assignment

# Runtime

input size $n$

① FF

runtime $= O(|f|(E+V)) = O(\underbrace{|f|}_{?} \cdot n^2)$

$|f| = \sum b_j$

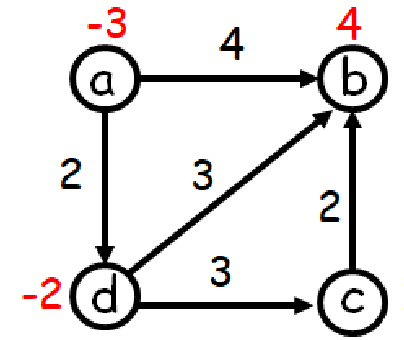
② EK

runtime $= O(V \cdot E^2)$

$= O(n^5)$

Break

# Circulation




Given a directed graph in which in addition to having capacities $c(u, v) \geq 0$ on each edge, we associate each vertex $v$ with a supply/demand value $d(v)$. We say that a vertex $v$ is a demand if $d(v) > 0$ and it is a supply if $d(v) < 0$.

We define a *circulation with demands* as a function $f: E \rightarrow \mathbb{R}^+$ that assigns nonnegative real values to the edges of $G$ and satisfies two axioms:

$0 \leq f(e) \leq c(e)$

1. Capacity constraint:
2. Conservation constraint: $f^{in}(v) - f^{out}(v) = \underbrace{d(v)}_{\text{new}} = \begin{cases} + \\ 0 \\ - \end{cases}$

# Necessary Condition

$\exists$ circulation $\Rightarrow \sum d = 0$

For every feasible circulation $\sum_{v \in V} d(v) = 0$

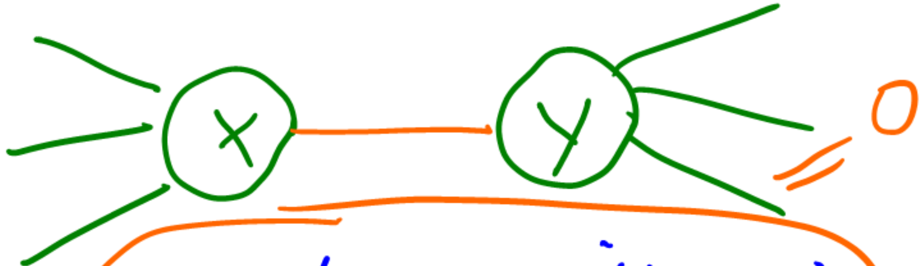
*Proof.*

start with a definition:

$$f^{in}(v) - f^{out}(v) = d(v)$$

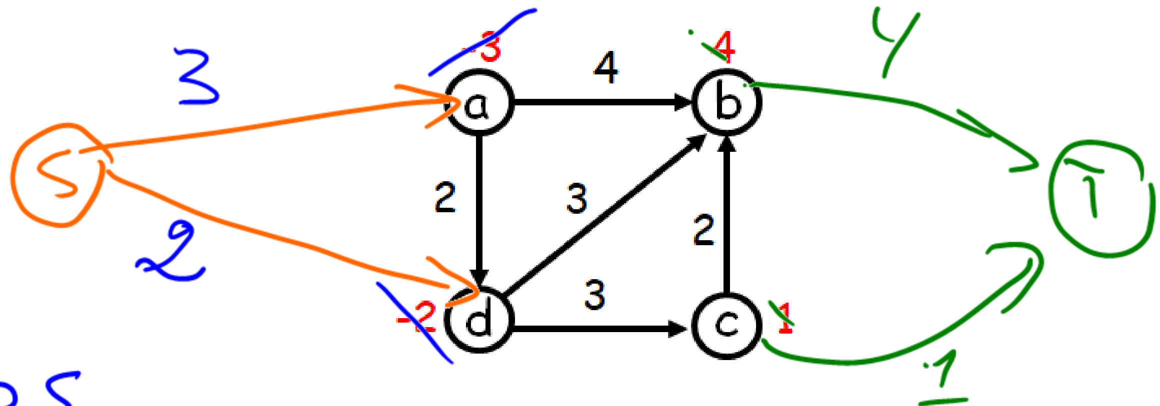
sum up over all vertices

$$0 = \sum_{v \in V} \left[ f^{in}(v) - f^{out}(v) \right] = \sum_{v \in V} d(v)$$



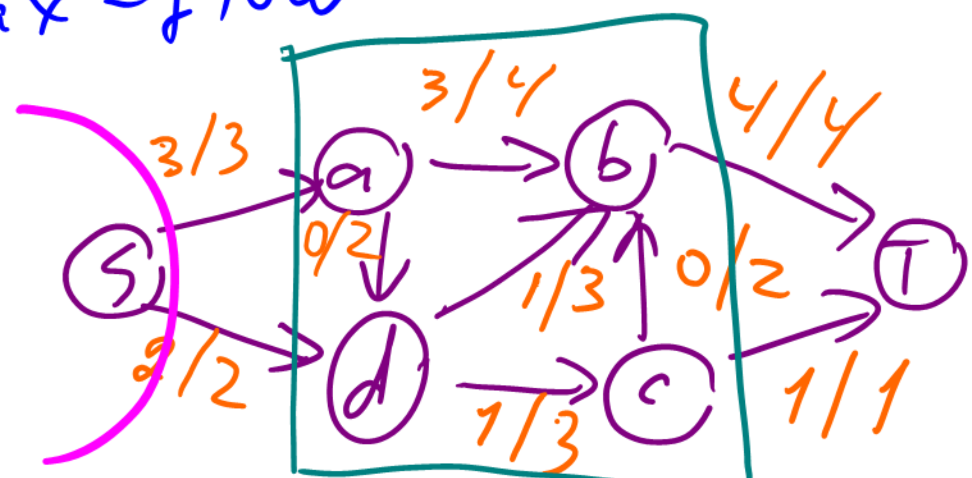

$$\underbrace{f^{in}(x) - f^{out}(x)}_{v=x} + \underbrace{f^{in}(y) - f^{out}(y)}_{v=y}$$

# Reduction to Flow Problem




Steps

1. Create a NF
2. Run FF (or any other algorithm)
3. Translate a max-flow back to circulation



# Circulation with Demands

<u>Claim:</u> There is a feasible circulation with demands d(v) in G if and only if the maximum s-t flow in G' has value D.

$$\sum_{d(v)>0} d(v) = D$$

Proof.

$\Rightarrow$) $\exists$ circulation, then max-flow = D

examine your construction

$\Leftarrow$) Given a max-flow = D

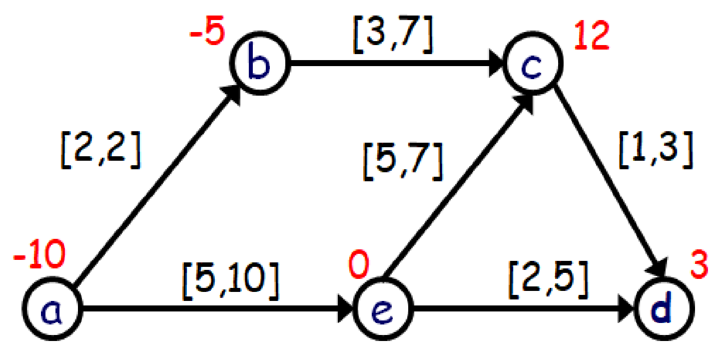
Find a circulation.

Take a final residual graph, remove S and T. Circulation is what is left.

# Circulation with Demands and Lower Bounds

We are given a directed graph $G=(V, E)$ with a capacity $c(e)$ and a lower bound $0 \le \ell(e) \le c(e)$ on each edge and a demand $d(v)$ on each vertex.

$[\ell(e), c(e]$




We define a *circulation with demands and lower bounds* as a function $f: E \to \mathbb{R}^+$ that assigns nonnegative real values to the edges of $G$ and satisfies two axioms:

$\ell(e) \le f(e) \le c(e)$

new

1. Capacity constraint:
2. Conservation constraint: $f^{in}(v) - f^{out}(v) = d(v)$

# Circulation with Demands and Lower Bounds

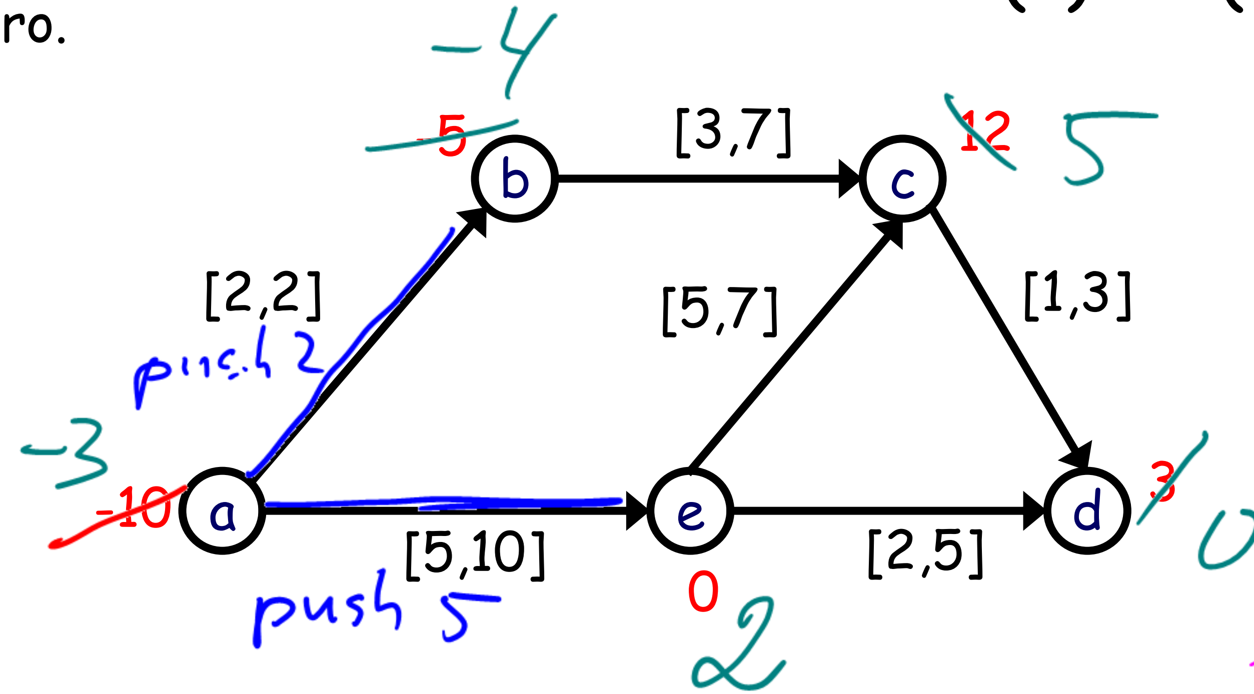
CDLB ≤ CD ≤ NF

First, we remove lower bounds and make all of them zero.

$$L(v) = f_0^{in}(v) - f_0^{out}(v)$$

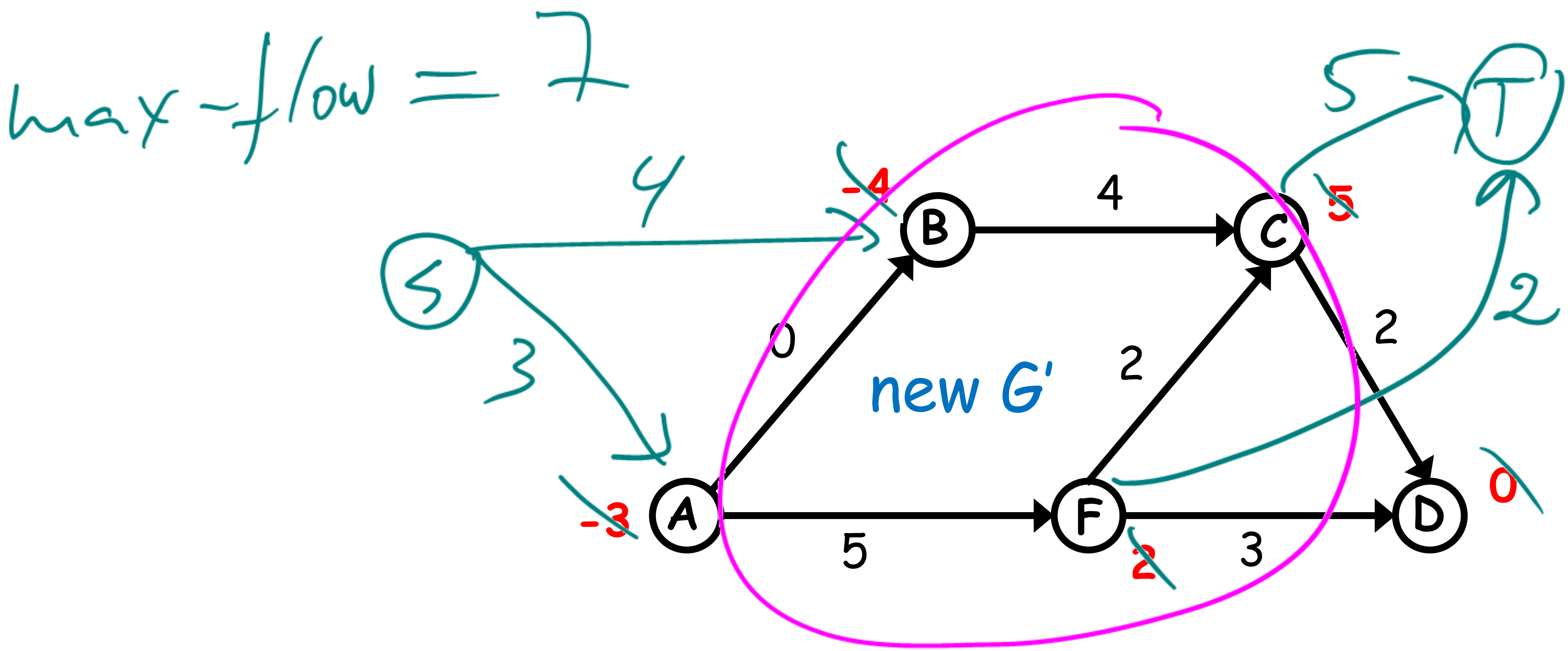
$$d'(v) = d(v) - L(v).$$

?




$$d'(b) = -5 + (-2) + 3 = -4$$

# Circulation with Demands and Lower Bounds




<u>Claim:</u> there is a feasible circulation in $G$ iff there is a feasible circulation in a new graph $G'$.

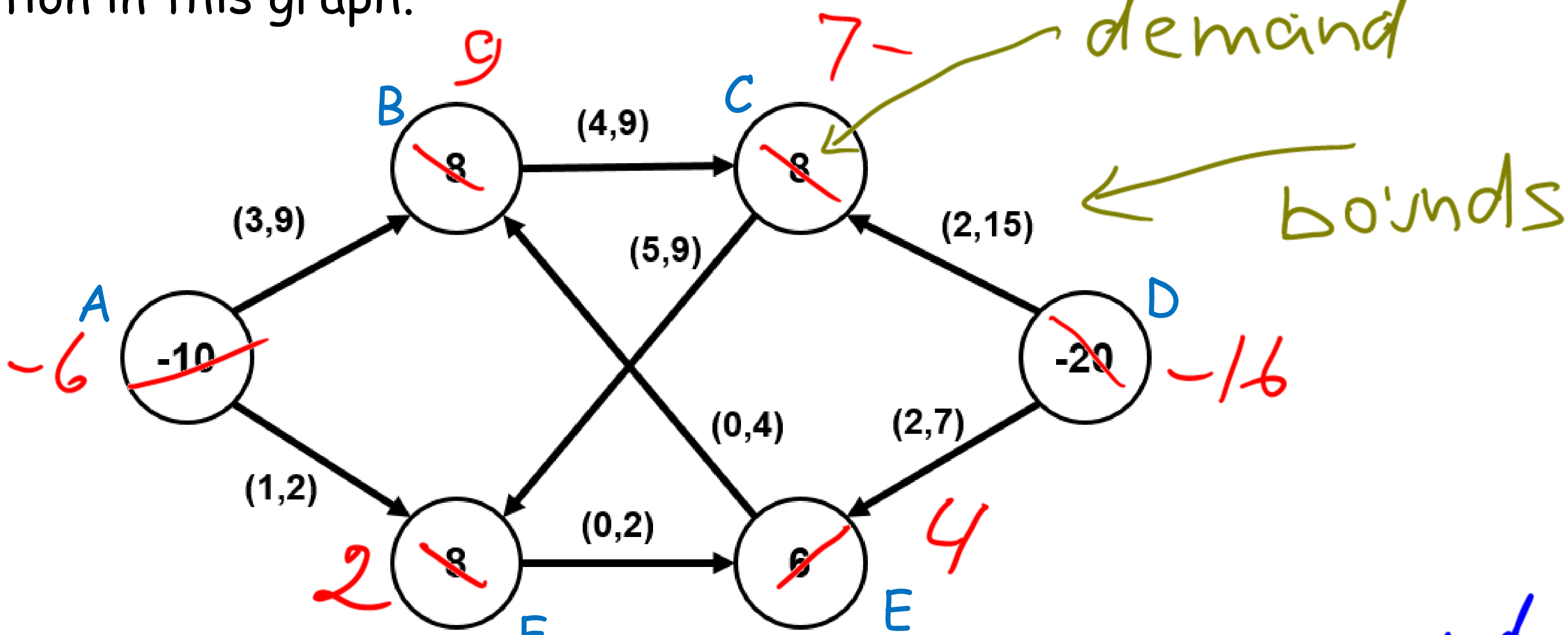
# Circulation with Demands and Lower Bounds

Summary: *given G with lower bounds, we:*

1. subtract lower bound $\ell(e)$ from the capacity of each edge.
2. subtract $L(v)$ from the demand of each node.
3. solve the circulation problem on this new graph to get a flow $f$.
4. add $\ell(e)$ to every $f(e)$ to get a flow for the original graph.

# Discussion Problem 3

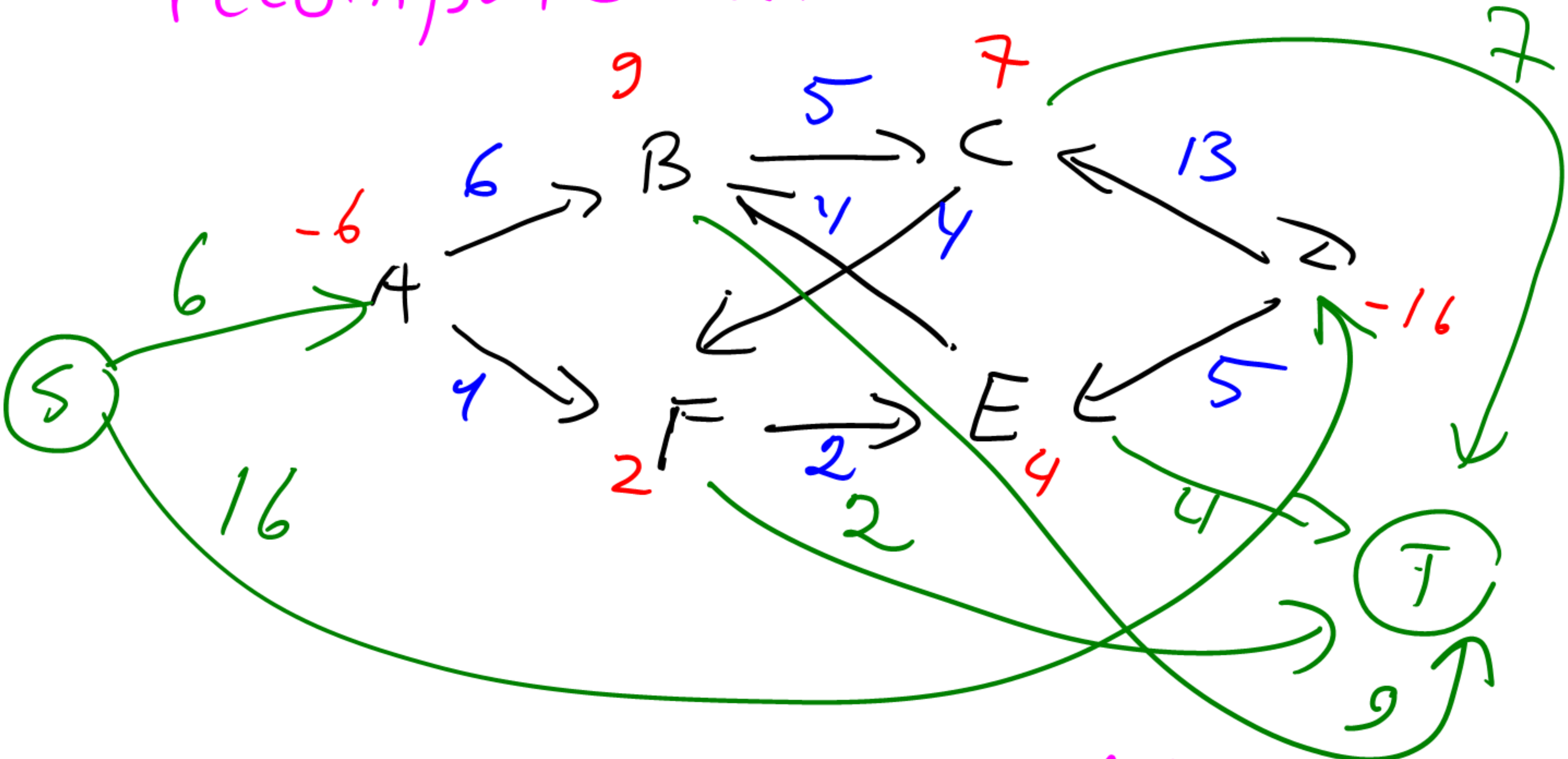
Given the network below with the demand values on vertices and lower bounds on edge capacities, determine if there is a feasible circulation in this graph.




① START! check a necessary cond.

$-10+8+8-20+6+8=0$

② remove lower bounds
recompute demands




Claim: $\exists$ circulation iff
max-flow $= 6+16 = 22$