Analysis of Algorithms

V. Adamchik

Lecture 9

Fall 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

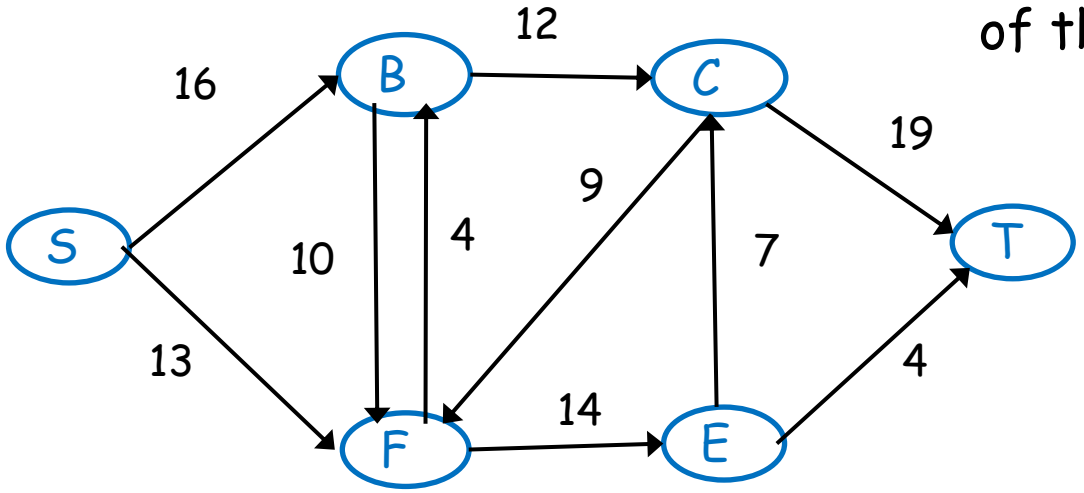
find a bottleneck

augment the flow along this path

update the residual graph G_f

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

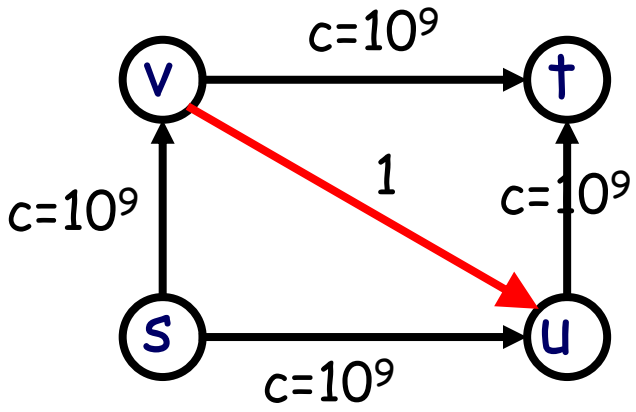
It is pseudo-polynomial because it depends on the size of the integers |f| in the input.



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

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 $O(V E^2)$.

(without proof)

Runtime history

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

year	discoverer(s)	bound
1951	Dantzig [11]	$O(n^2mU)$
1956	Ford & Fulkerson [17]	O(m U)
1970	Dinitz [13] Edmonds & Karp [15]	$O(n\,m^2)$ shortest path
1970	Dinitz [13]	$O(n^2m)$
1972	Edmonds & Karp [15] Dinitz [14]	$O(m^2 \log U)$ capacity scaling
1973	Dinitz [14] Gabow [19]	$O(nm \log U)$
1974	Karzanov [36]	$O(n^3)$ preflow-push
1977	Cherkassky [9]	$O(n^2m^{1/2})$
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
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

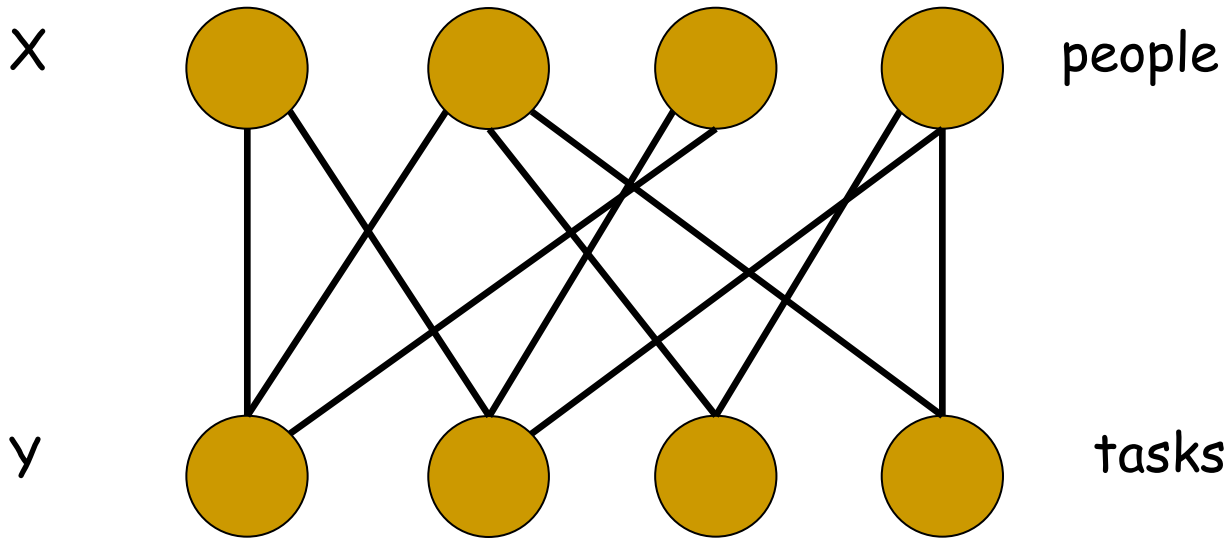
Formally, to reduce a problem Y to a problem X (we write $Y \leq_p X$) we want a function f that maps Y to X such that:

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

Solving by reduction to NF

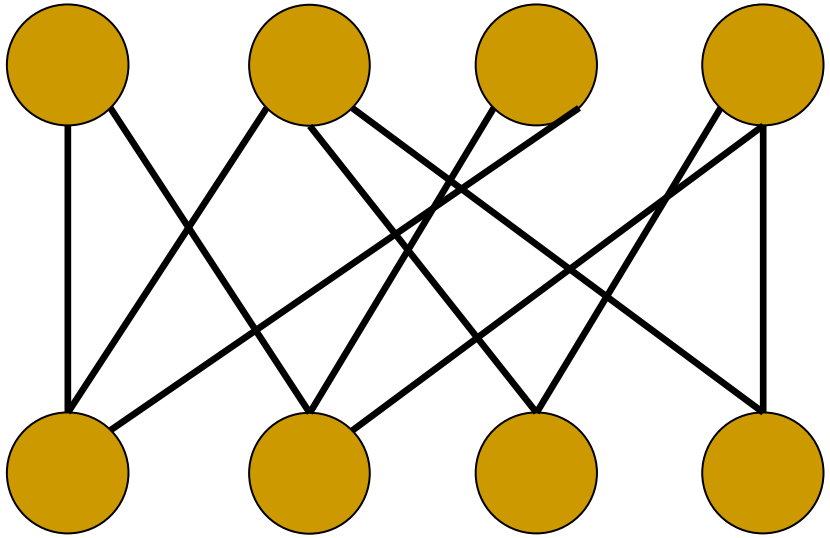
1. Describe how to construct a flow network.
2. Make a claim. Something like "this problem has a feasible solution if and only if the max flow is ...".
3. Prove the above claim in both directions.

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

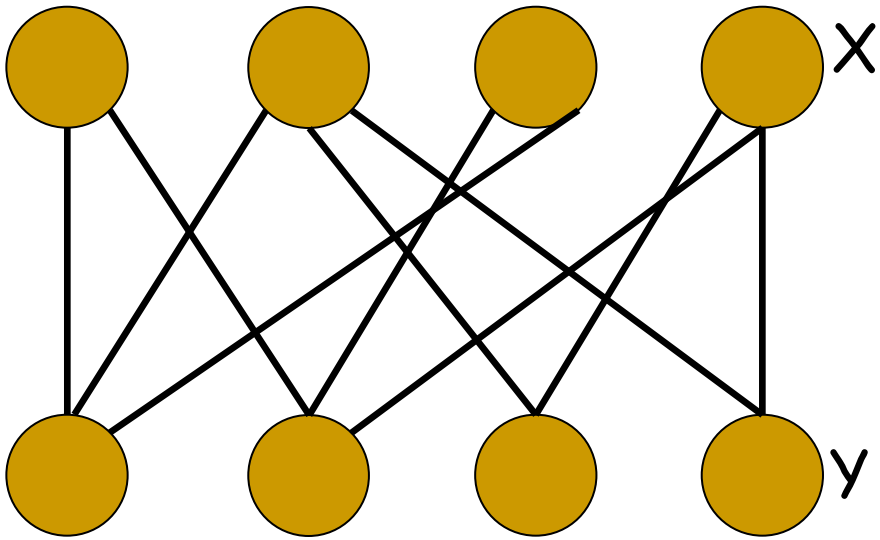
Given bipartite $G = (X, Y, E)$. Let $|X|=|Y|=V$.

$\forall e \in E$, 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.

Max matching $\Longrightarrow$ Max flow

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

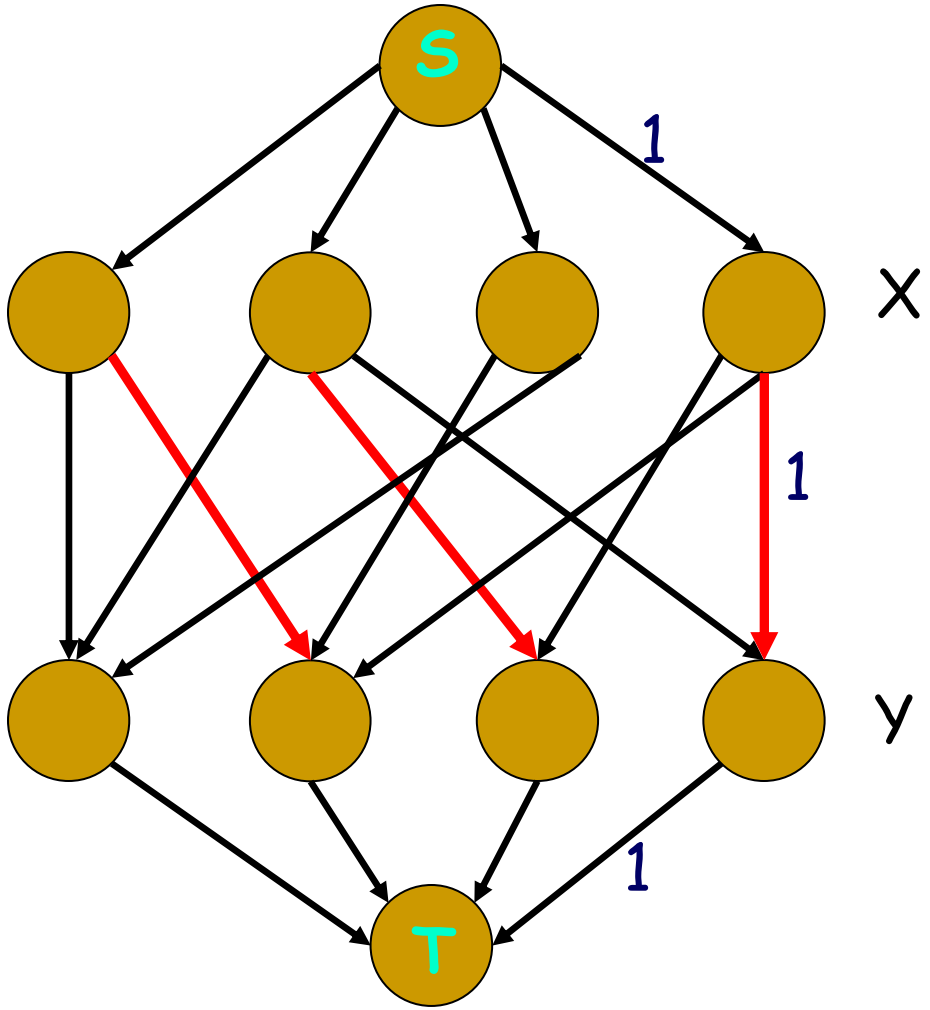
Proof.

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.



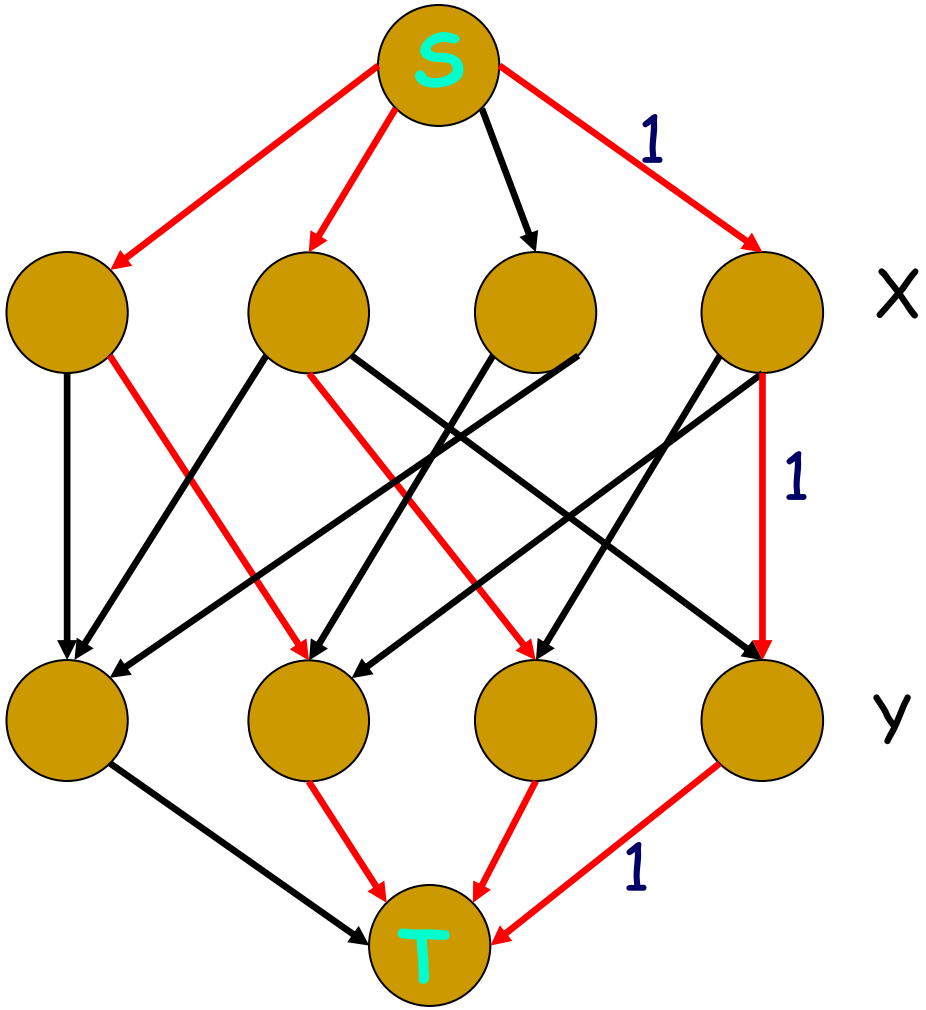
Max matching ⇐ Max flow

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

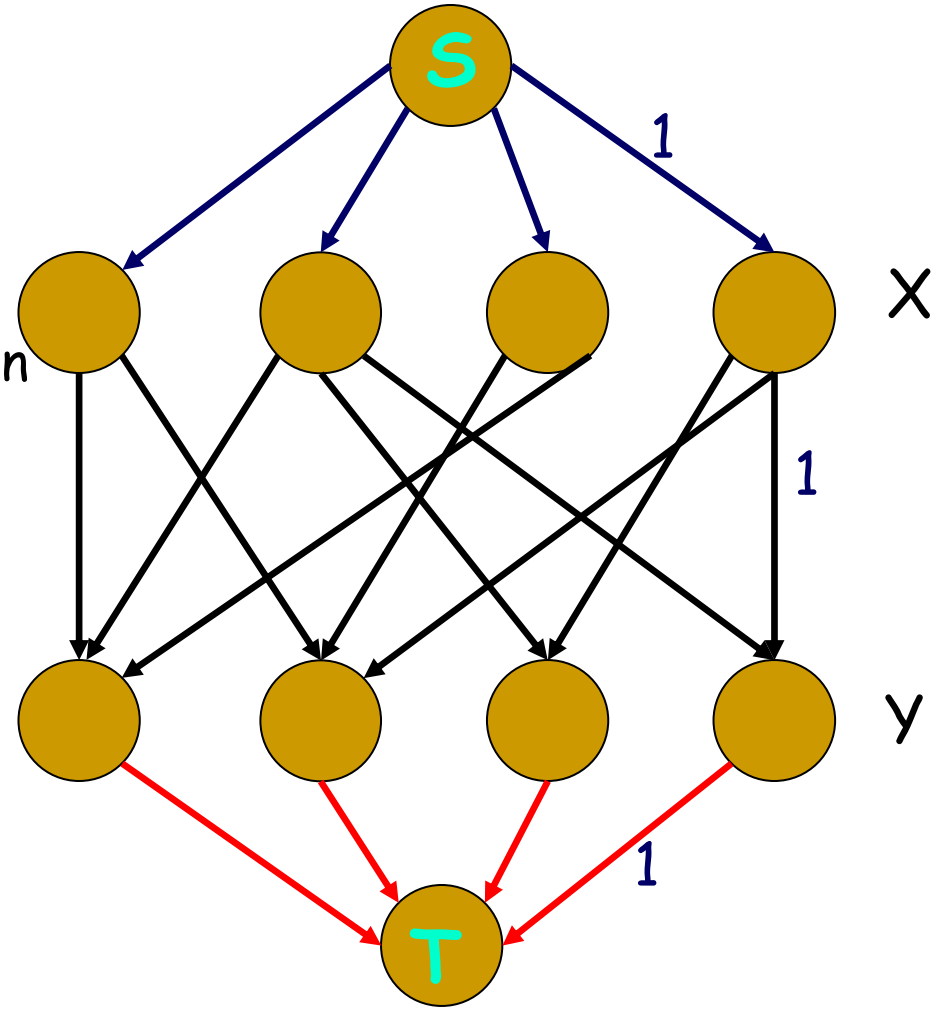
$$|f| = \sum_{e \text{ out of } X} f(e) - \sum_{e \text{ in to } X} f(e)$$



Runtime Complexity

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

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



The running time of Ford-Fulkerson is
$O(|f| (E' + V'))$

where

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

$V' = 2V + 2$

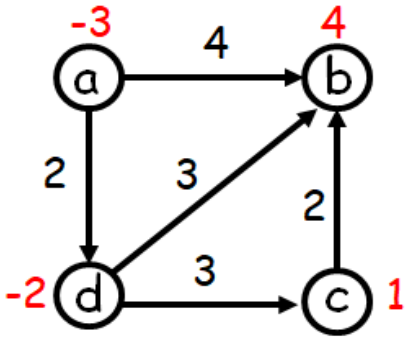
$E' = E + 2V$.

So, the runtime is $O(V (E + 2V + 2V + 2)) = O(V E + V^2) = O(V E)$

Discussion Problem 1

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.

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:

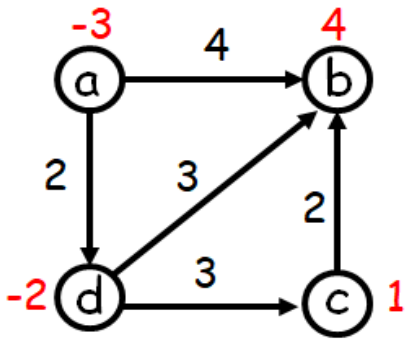
1. Capacity constraint:
2. Conservation constraint:

Necessary Condition

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

Proof.

Reduction to Flow Problem



Circulation with Demands

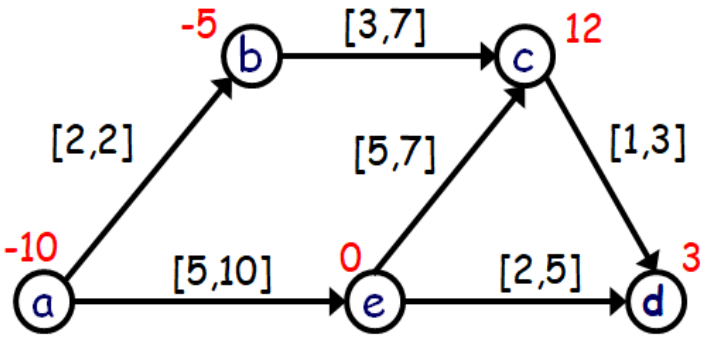
Claim: There is a feasible circulation with demands d(v) in G if and only if the maximum s-t flow in G' has value D.

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

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:

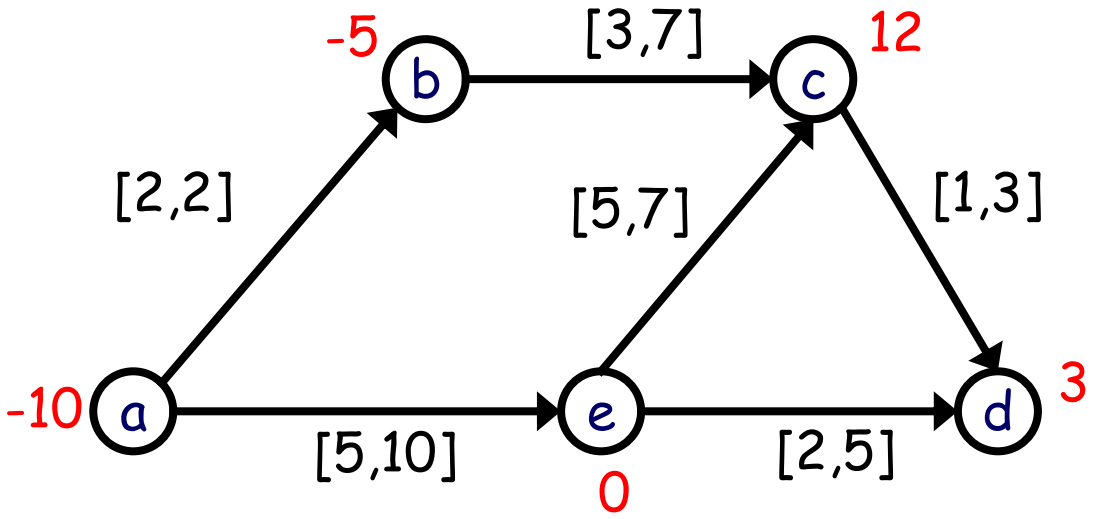
1. Capacity constraint:
2. Conservation constraint:

Circulation with Demands and Lower Bounds

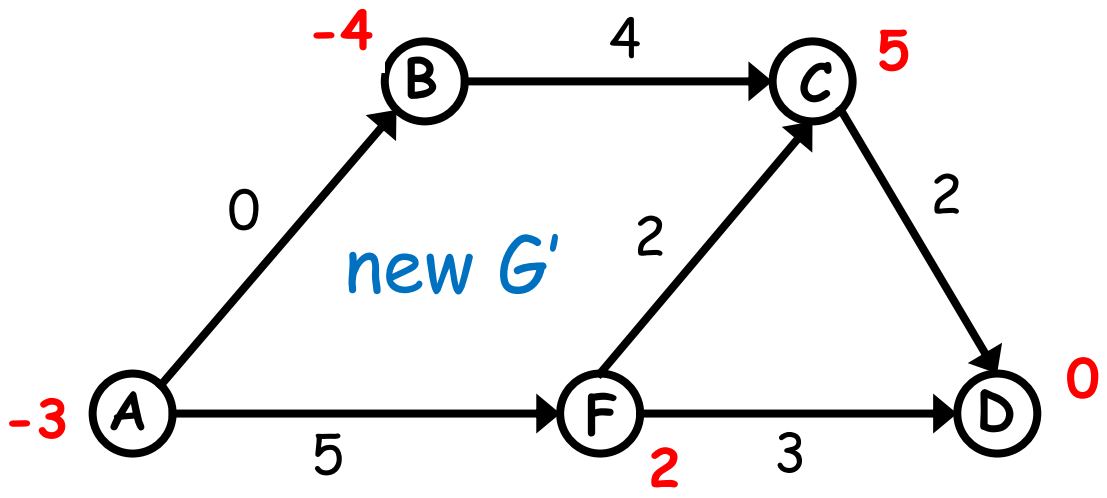
First, we remove lower bounds and make all of them zero.

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

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



Circulation with Demands and Lower Bounds



Claim: there is a feasible circulation in G iff there is a feasible circulation in a new graph G'.

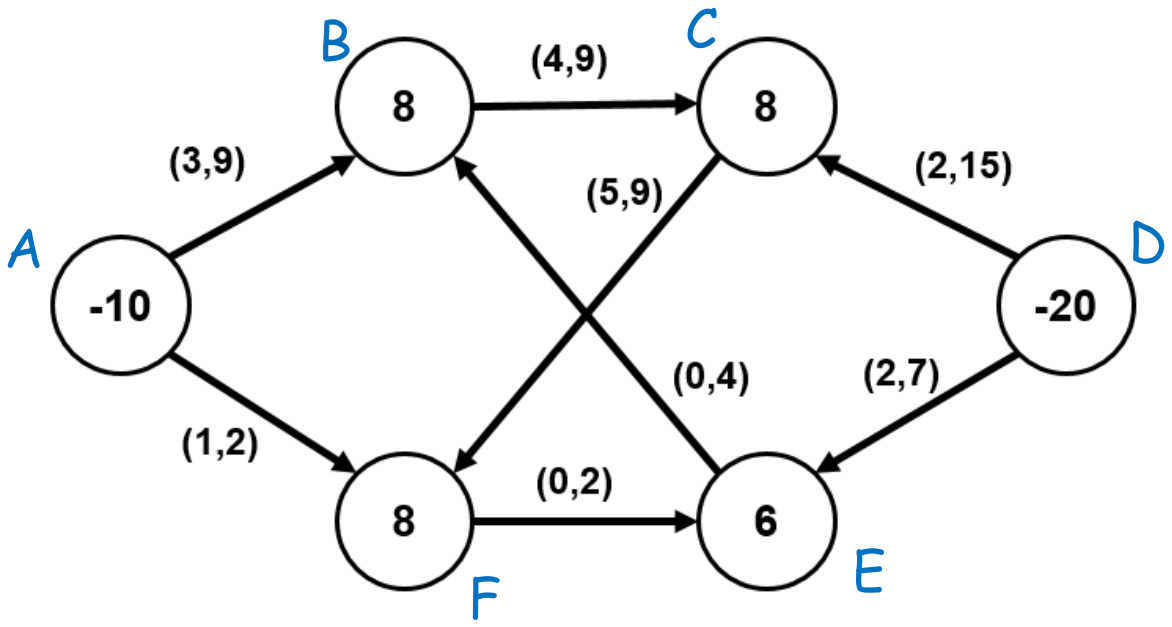
Circulation with Demands and Lower Bounds

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 2

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.



Discussion Problem 3

CSCI 570 is a large class with n TAs. Each week TAs must hold office hours in the TA office room. There is a set of k hour-long time intervals $I_1, I_2, \ldots I_k$ in which the office room is available. The room can accommodate up to 3 TAs at any time. Each TA provides a subset of the time intervals he or she can hold office hours with the minimum requirement of l_j hour per week, and the maximum m_j hours per week. Lastly, the total number of office hours held during the week must be H. Design an algorithm to determine if there is a valid way to schedule the TA's office hours with respect to these constraints.

Discussion Problem 4

The computer science department course structure is represented as a directed acyclic graph $G = (V, E)$ where the vertices correspond to courses and a directed edge (u, v) exists if and only if the course u is a prerequisite of the course v. By taking a course w, you gain a benefit of p_w which could be a positive or negative number. Note, to take a course, you have to take all its prerequisites. Design an efficient algorithm that picks a subset $S \subset V$ of courses such that the total benefit is maximized.

benefit = $\sum p_w$, where $w \in S$.

goal: max benefit