Analysis of Algorithms

V. Adamchik

CSCI 570

Lecture 8

University of Southern California

Fall 2023

Network Flow

Reading: chapter 7.1 - 7.4

The Network Flow Problem

Our fourth major algorithm design technique

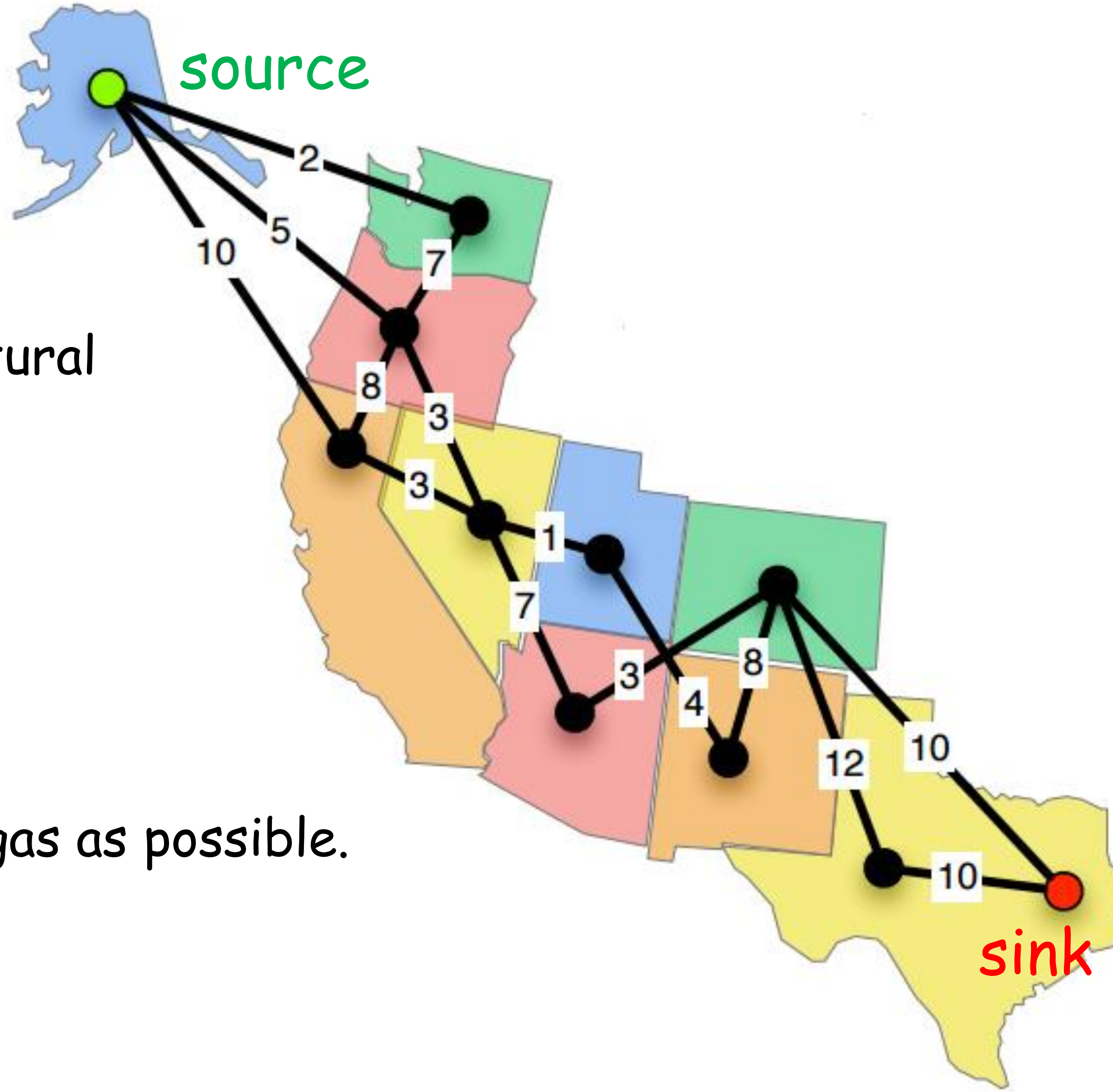
(greedy, divide-and-conquer, and dynamic programming).

Plan:

The Ford-Fulkerson algorithm

Max-Flow Min-Cut Theorem

The Flow Problem



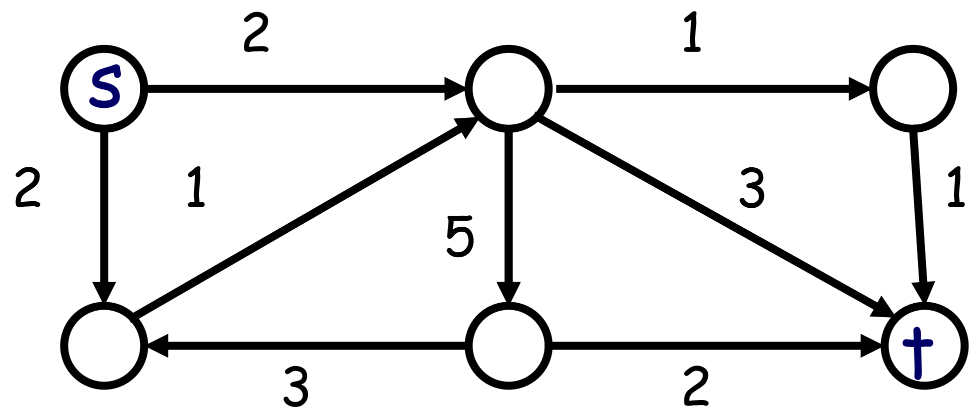
Suppose you want to ship natural gas from Alaska to Texas.

Pipes have capacities.

The goal is to send as much gas as possible.

How can you do it?

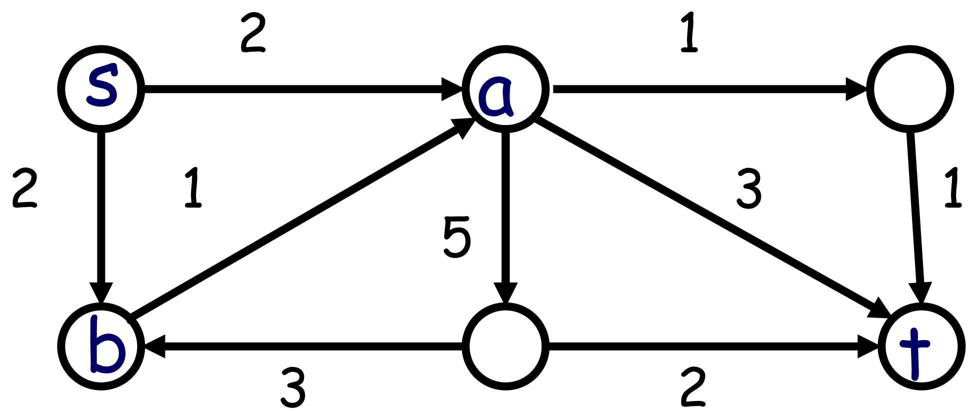
The Max-Flow Problem



we define a *flow* 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:

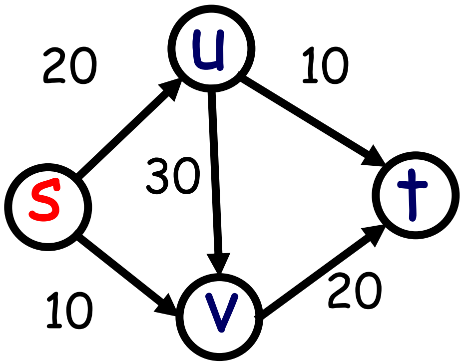
The MAX Flow Problem



The max-flow here is __.

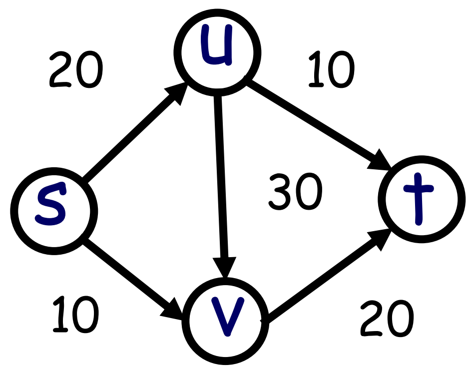
How can you see that the flow is really max?

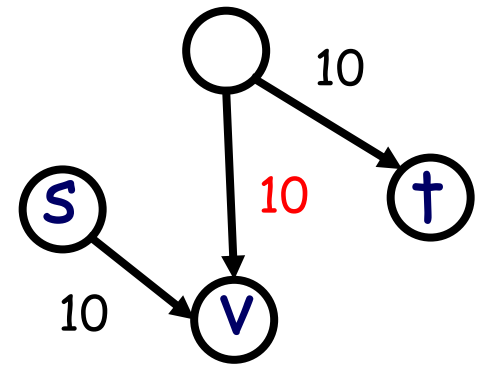
Greedy Approach: push the max



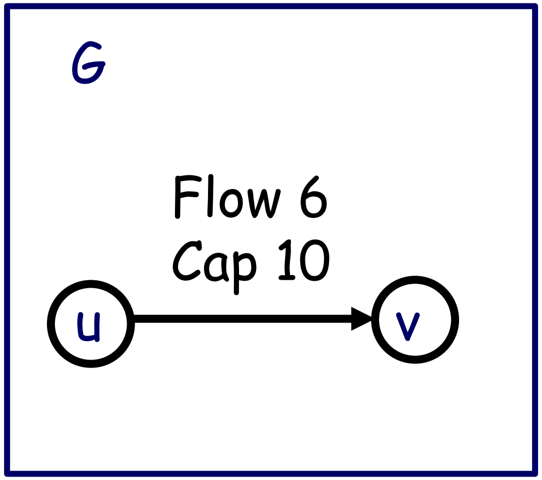
Canceling Flow

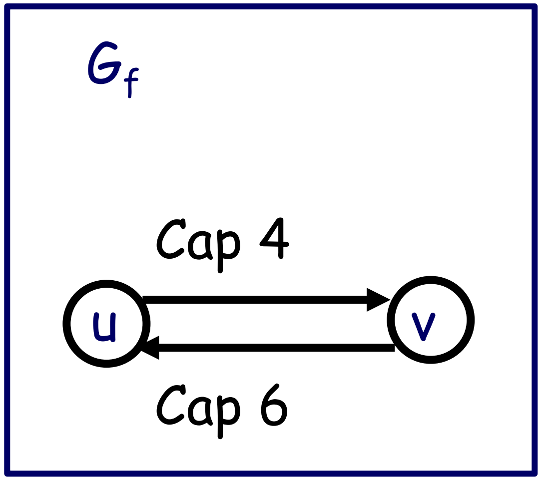
Push 20 via s-u-v-t



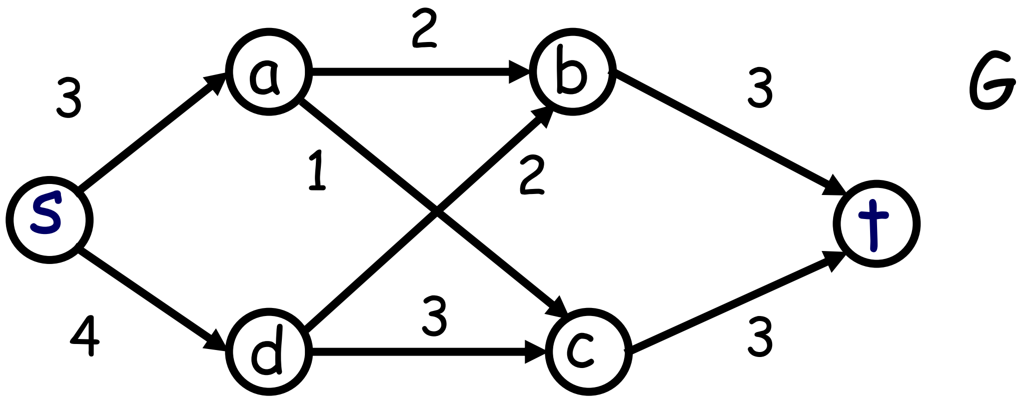


Residual Graph G_f
Residual Capacity c_f





Example: residual graph



Push 2 along s-d-b-t and draw the residual graph

Augmenting Path = Path in G_f

Let P be an s–t path in the residual graph G_f.

Let bottleneck(P) be the smallest capacity in G_f on any edge of P.

If bottleneck(P) > 0 then we can increase the flow by sending bottleneck(P) units of flow along the path P.

augment(f, P):
b = bottleneck(P)
for each $e = (u,v) \in P$:
    if e is a forward edge:
        decrease $c_f(e)$ by b //add some flow
    else:
        increase capacity by b //erase some flow

The Ford-Fulkerson Algorithm

<u>Algorithm</u>. Given $(G, s, t, c \in \mathbb{N}^+)$

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

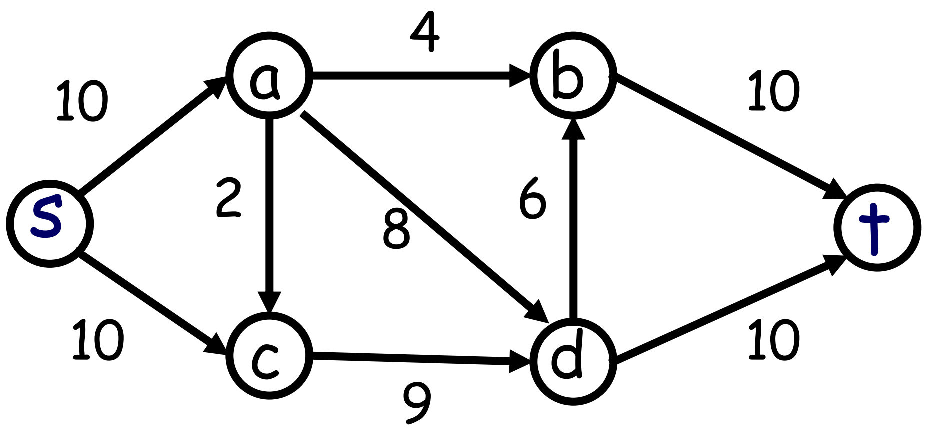
while exists an augmenting path in G_f

    find bottleneck

    augment the flow along this path

    update the residual graph G_f

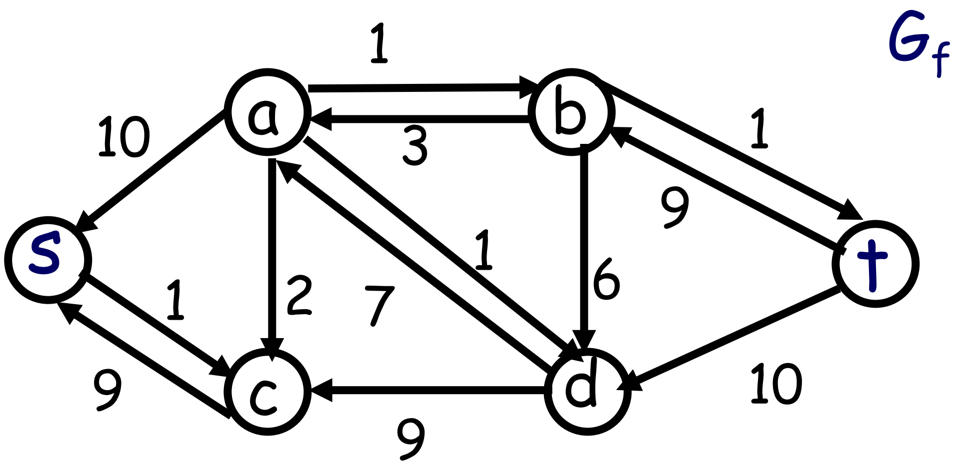
Example



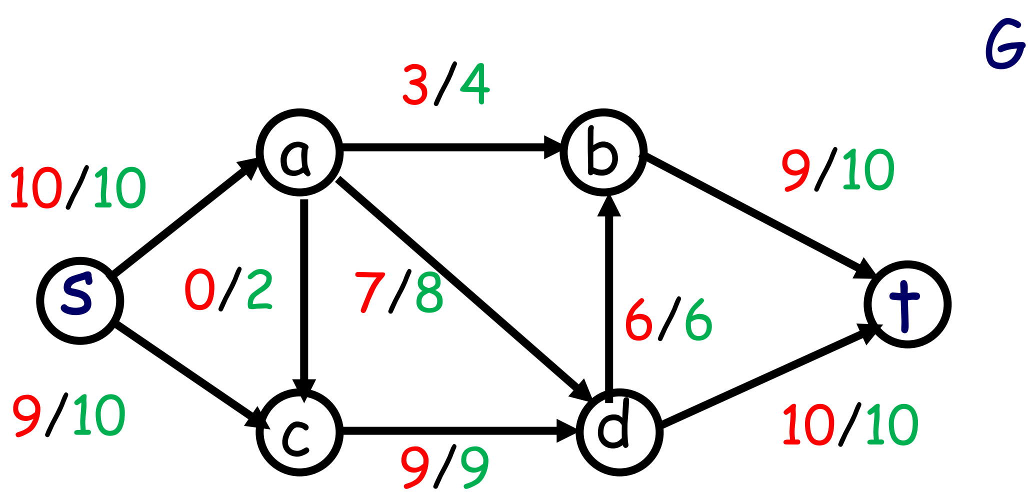
Path s-a-c-d-t

Example

Path s-c-a-b-t



In graph G edges are with flow/cap notation



The Ford-Fulkerson Algorithm

Runtime Complexity

Algorithm. Given (G, s, t, $c \in \mathbb{N}^+$)

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

while exists an augmenting path in G_f

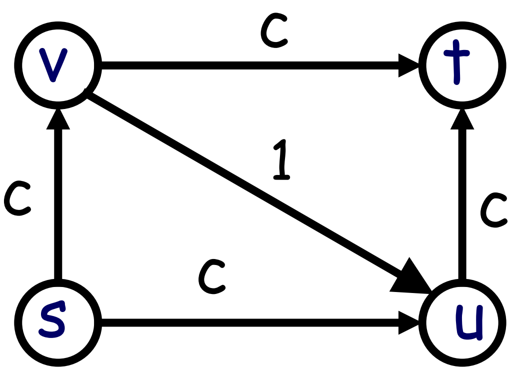
find bottleneck

augment the flow along this path

update the residual graph

The worst-case

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

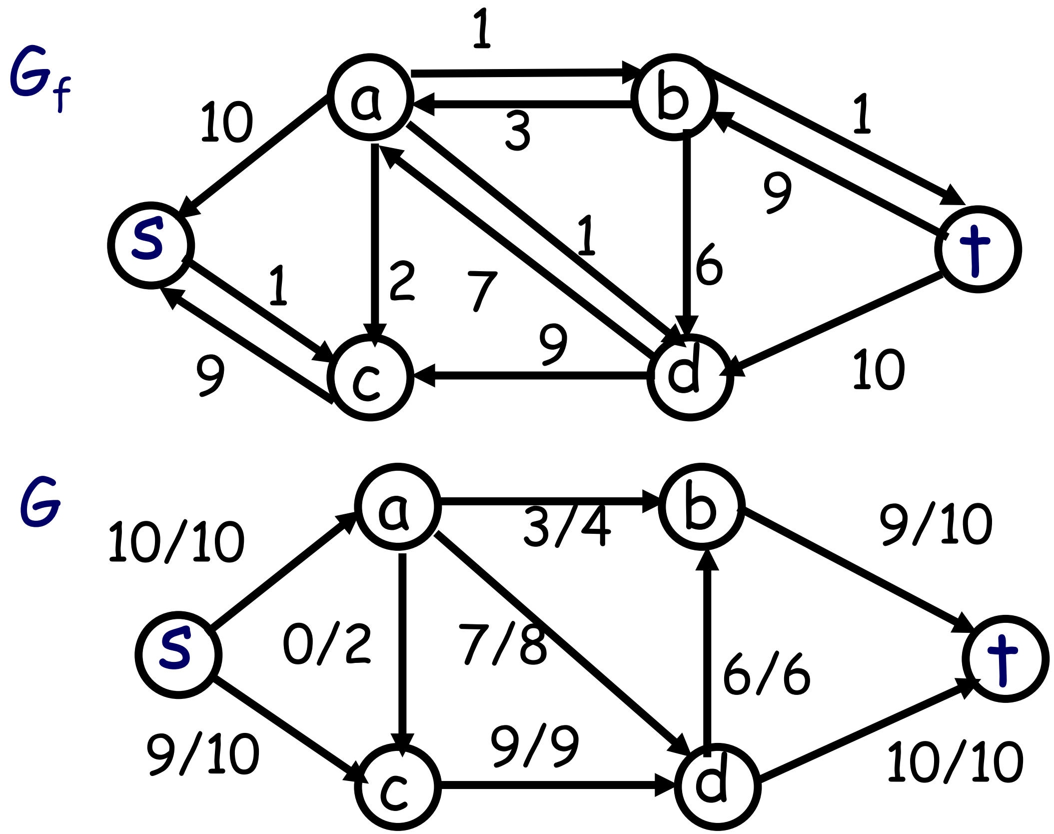


$c=10^9$

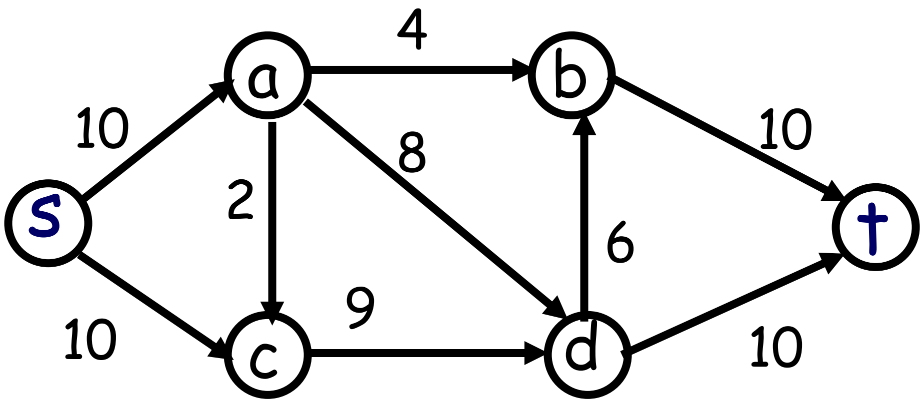
Proof of Correctness

How do we know the algorithm terminate

How do we know the flow is maximum?

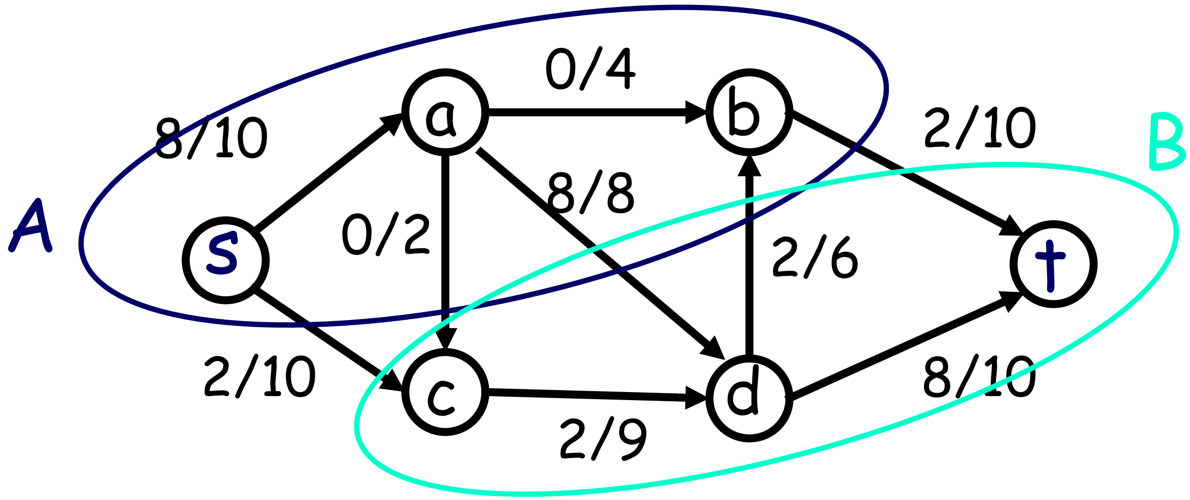


Cuts and Cut Capacity



Cuts and Flows

Consider a graph with some flow and cut



The flow-out of A is ___

The flow-in to A is ___

The flow across (A,B) is ___

What is a flow value |f| in this graph?

Lemma 1

For any flow f and any (A,B) cut

$$|f| = \sum_{v} f(s,v) = \sum_{u \in A, v \in B} f(u,v) - \sum_{u \in A, v \in B} f(v,u)$$

Proof.

Lemma 2

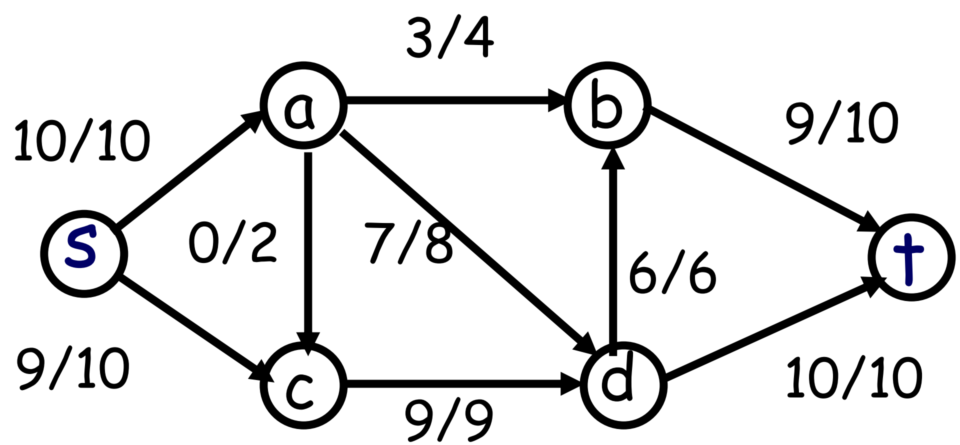
For any flow f and any (A,B) cut $|f| \leq cap(A,B)$.

Proof.

Max-flow Theorem

<u>Theorem</u>. *The Ford-Fulkerson algorithm outputs the maximum flow.*

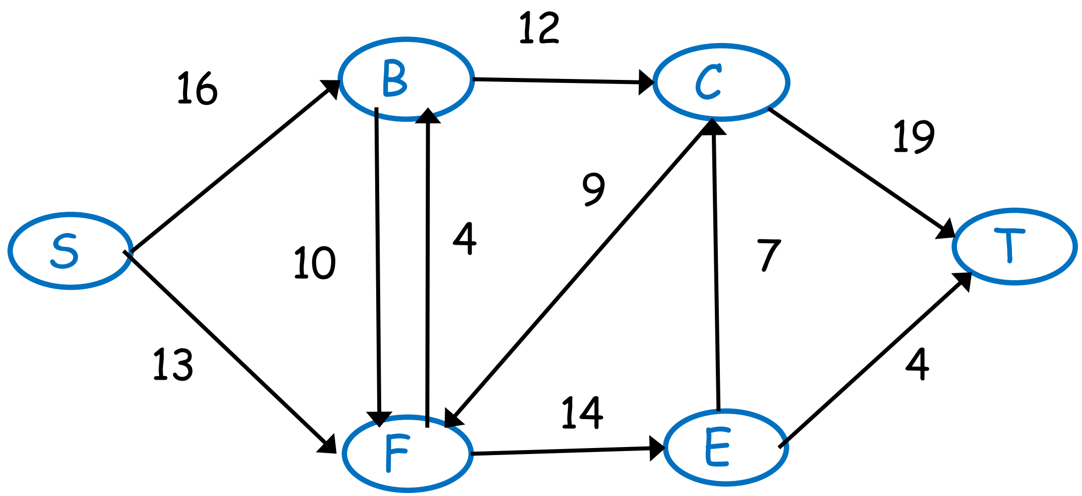
$$\max_{f} |f| = \min_{(A,B)} cap(A, B)$$



Where is a min-cut ?

Discussion Problem 1

Run the Ford-Fulkerson algorithm on the following network:

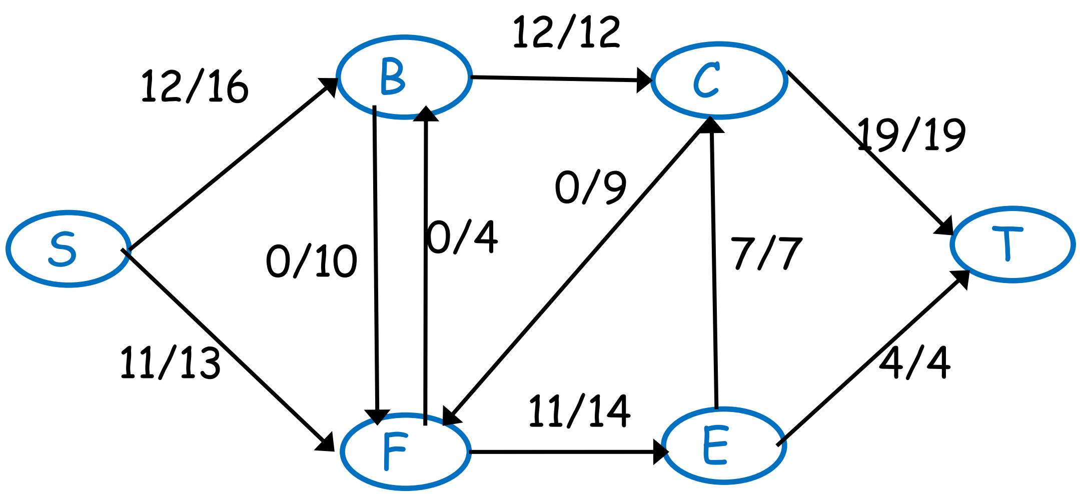


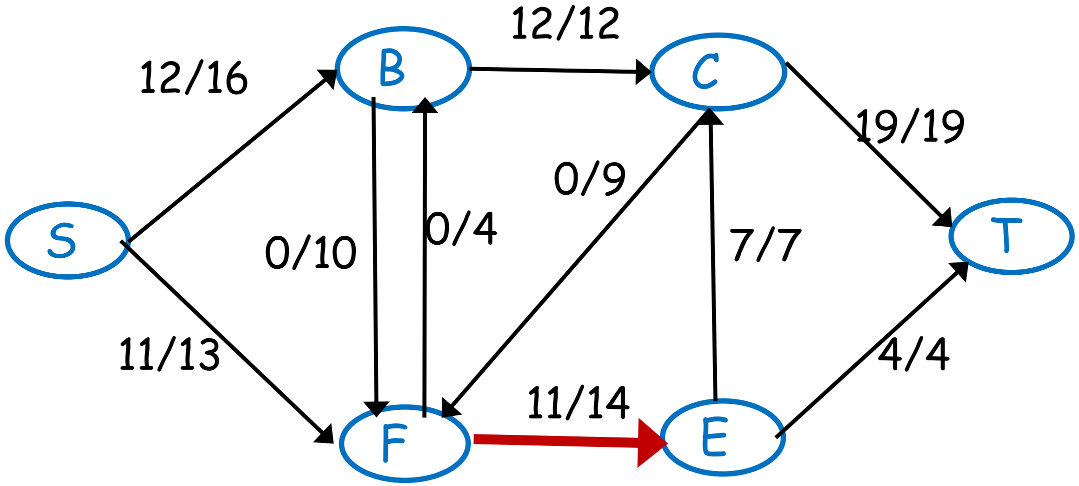
How do you find a min-cut ?

Is a min-cut unique?

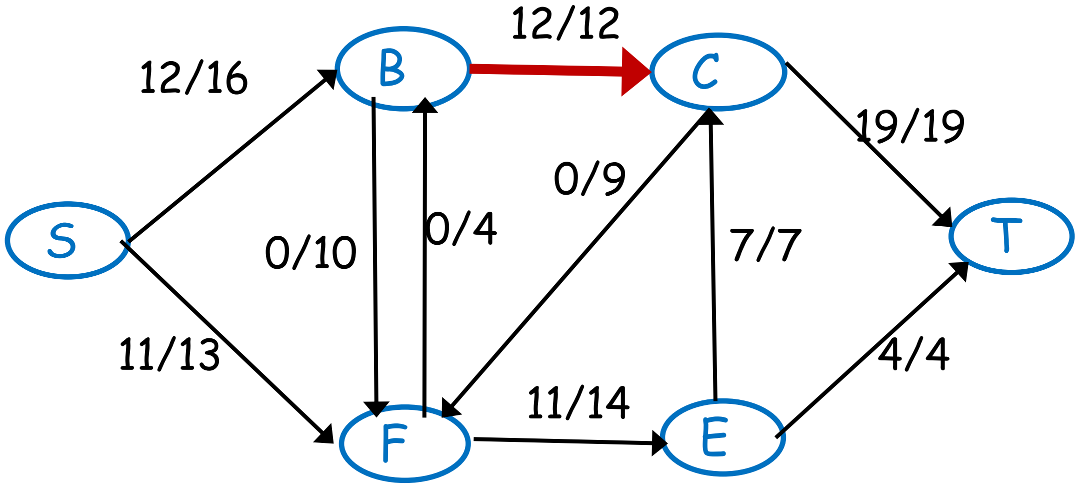
Discussion Problem 2

You have successfully computed a maximum s-t flow for a network $G = (V, E)$ with positive integer edge capacities. Your boss now gives you another network G' that is identical to G except that the capacity of exactly one edge is decreased by one. You are also explicitly given the edge whose capacity was changed. Describe how you can compute a maximum flow for G' in linear time.





12/12
12/16
B
C
19/19
0/9
S
0/10
0/4
7/7
T
11/13
4/4
11/14
F
E



S
B
C
F
E
T
12/16
12/12
19/19
0/10
0/4
0/9
7/7
11/13
11/14
4/4

Discussion Problem 3

If we add the same positive number to the capacity of every directed edge, then the minimum cut (but not its value) remains unchanged. If it is true, prove it, otherwise provide a counterexample.

Discussion Problem 4

In a daring burglary, someone attempted to steal all the candy bars from the CS department. Luckily, he was quickly detected, and now, the course staff and students will have to keep him from escaping from campus. In order to do so, they can be deployed to monitor strategic routes. Compute the minimum number of students/staff needed and show the monitored routes.

Reduction

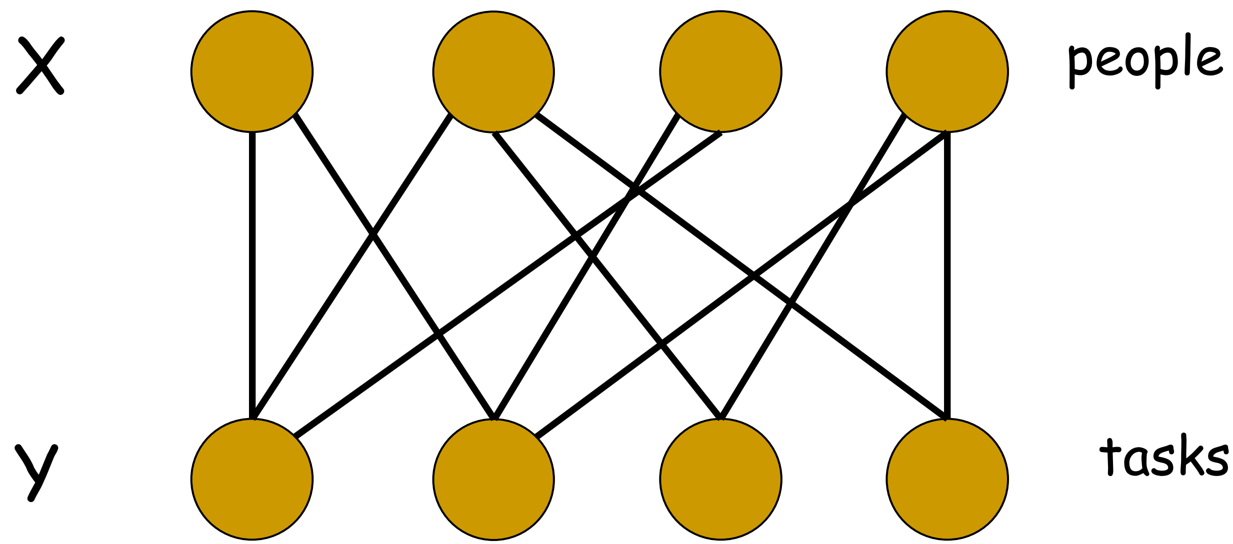
Formally, to reduce a problem Y to a problem X (we write $Y \leq_p X$) we want a function f that maps Y to X such that:

- 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

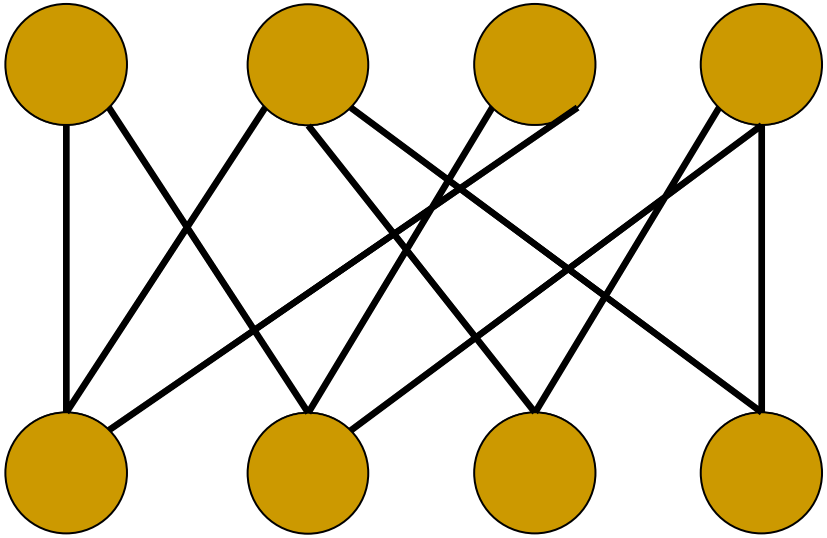
1. Describe how to construct a flow network
2. Make a claim. Something like "this problem has a feasible solution if and only if the max flow is ..."
3. Prove the above claim in both directions

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.

Solving by Reduction

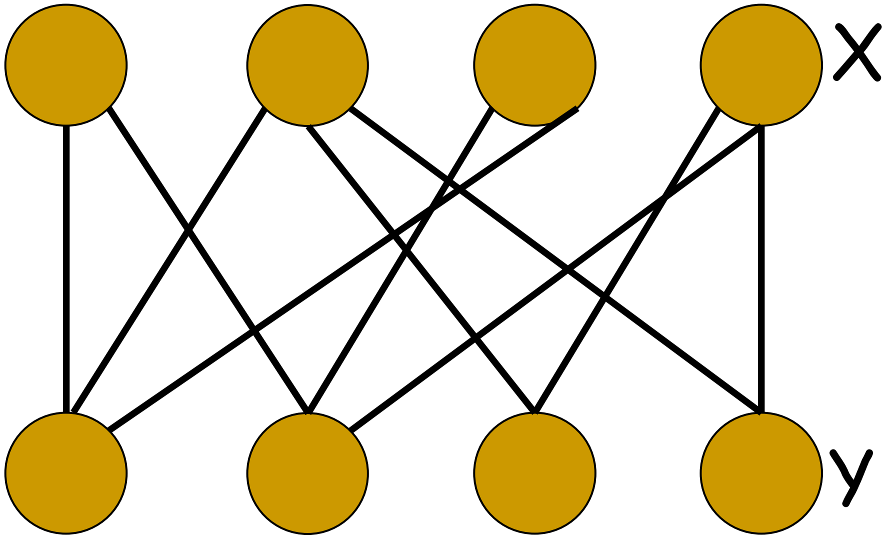
Given an instance of bipartite matching.

Create an instance of network flow.

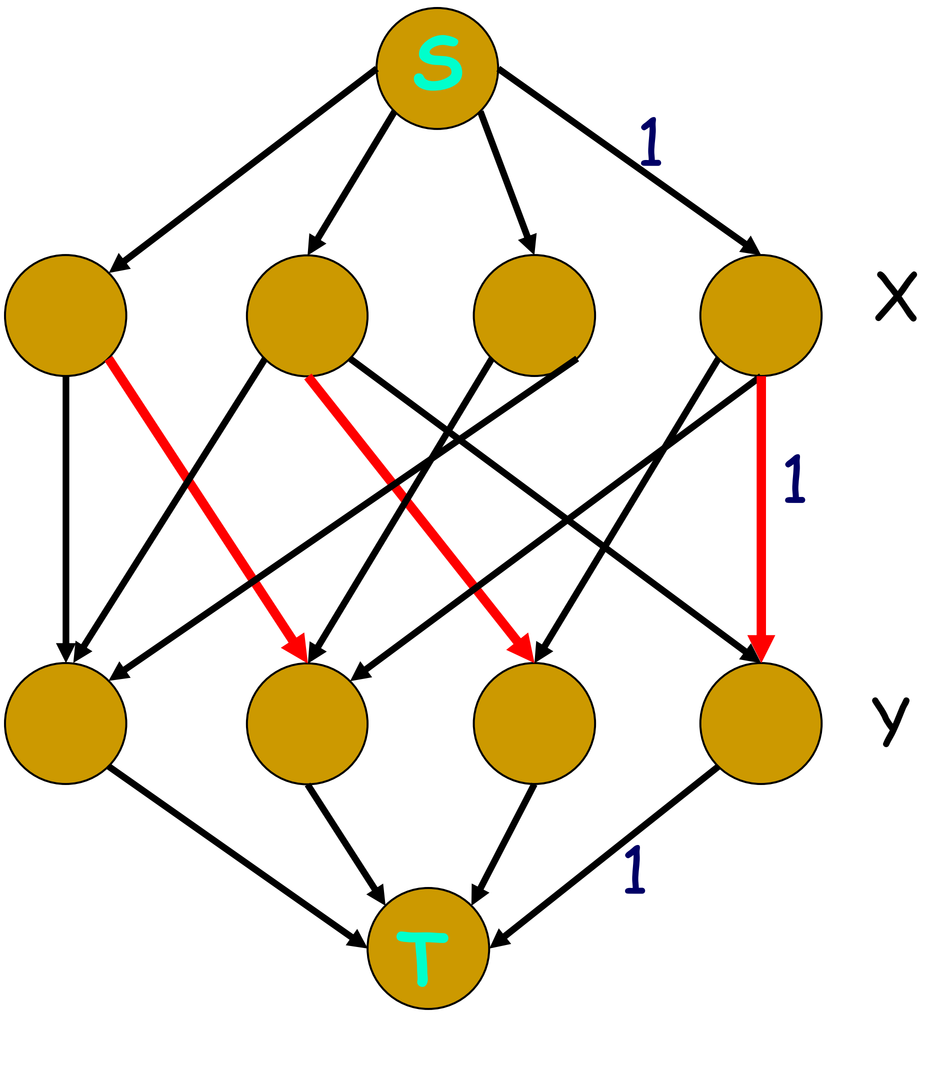
The solution to that network flow problem can easily be used to find the solution to the bipartite matching problem.

Reducing Bipartite Matching to Network Flow

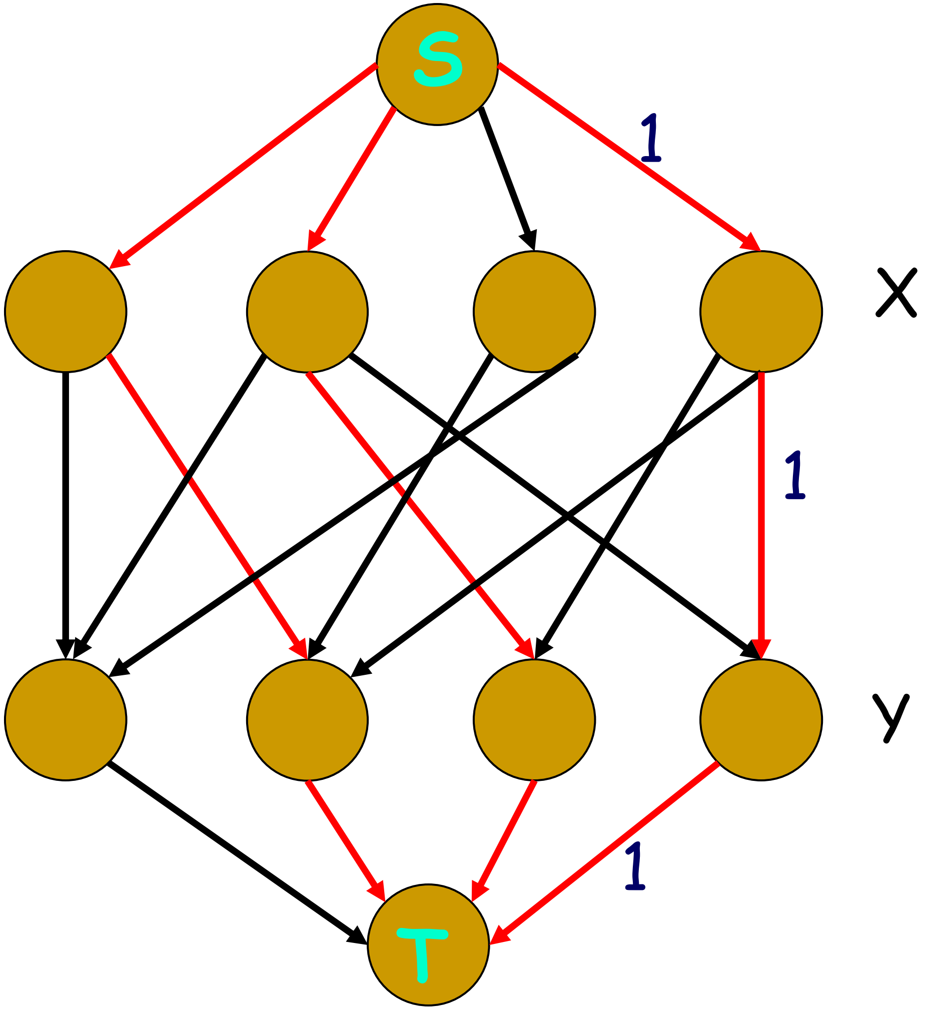
Given bipartite $G = (X, Y, E)$. Let $|X|=|Y|=V$.



Max matching = Max flow



Max matching = Max flow

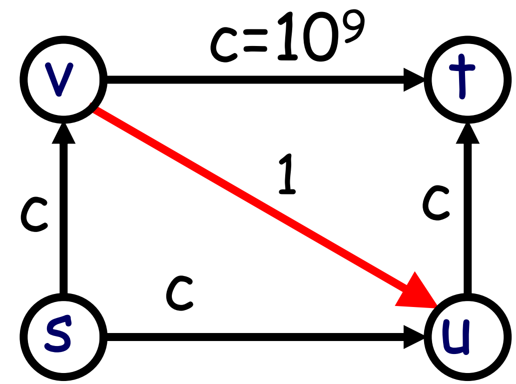


Runtime Complexity

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

How to improve the efficiency of the Ford-Fulkerson Algorithm?

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



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]	$O(n^2 mU)$	
1956	Ford & Fulkerson [17]	O(m U)	
1970	Dinitz [13] Edmonds & Karp [15]	O(n m²)	shortest path
1970	Dinitz [13]	$O(n^2 m)$	
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]	$O(n^2 m^{1/2})$	
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
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)	

Discussion Problem 6

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?