Analysis of Algorithms

V. Adamchik CSCI 570 Spring 2020

Lecture 9 University of Southern California

Network Flow

Reading: chapter 7

The Network Flow Problem

Our fourth major algorithm design technique (greedy, divide-and-conquer, and dynamic programming).

Input(X) $\longrightarrow$ Input(NF)

solve(NF)

Output(X) $\longleftarrow$ Output(NF)

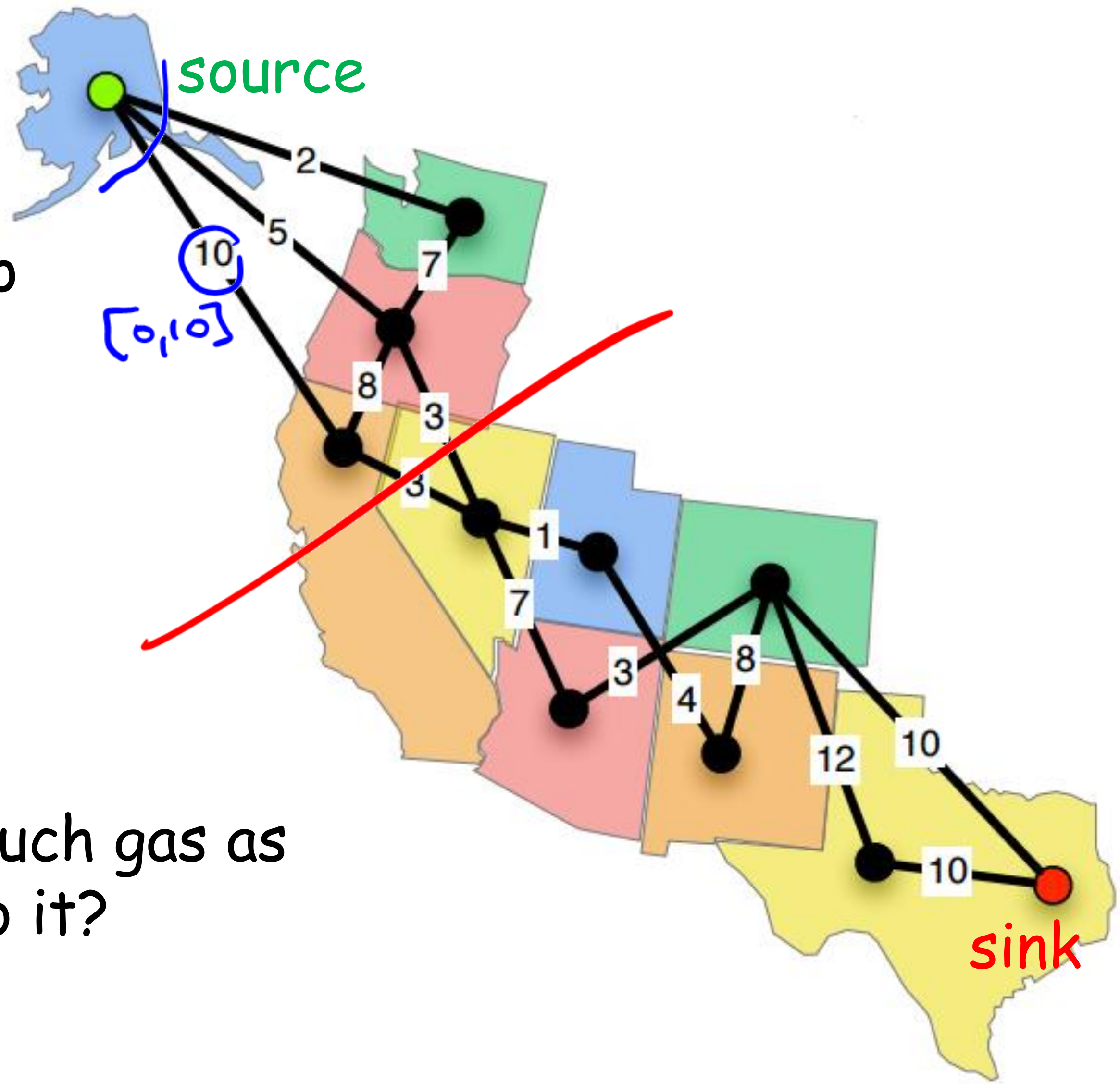
Plan:

FF

The Ford-Fulkerson algorithm

Max-Flow Min-Cut Theorem

The Flow Problem

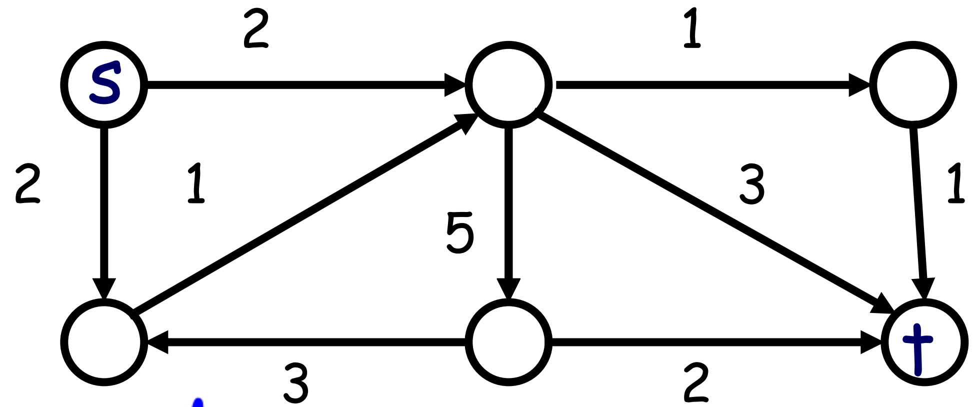


Suppose you want to ship natural gas from Alaska to Texas.

Pipes have capacities.

The goal is to send as much gas as possible. How can you do it?

The Max-Flow Problem



$NF = (V, E, s, t, c)$

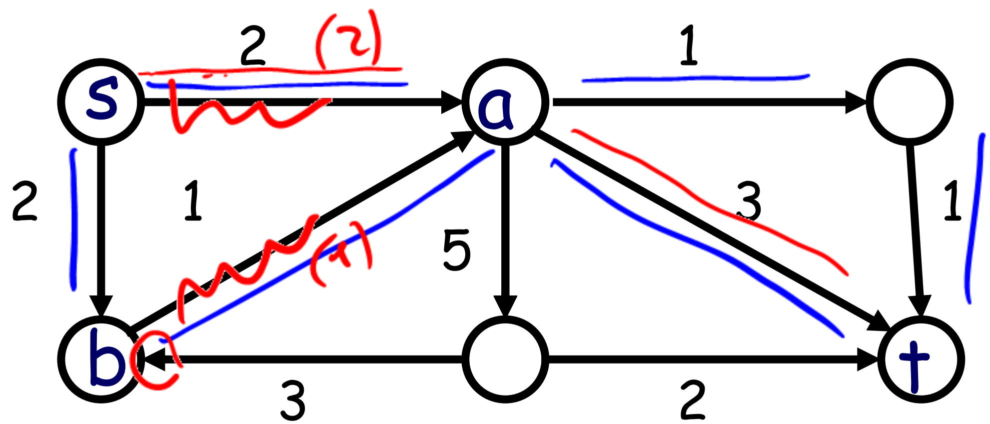
$c(e) \geq 0, \ e \in E$
$c(e) = 0, \ e \notin E$

Def. A flow $f(e)$

1) $0 \leq f(e) \leq c(e)$, capacity law

2) $\sum_{\text{into } v} f(e) = \sum_{\text{out of } v} f(e)$, conservation law $v \neq \{s, t\}$

The MAX Flow Problem

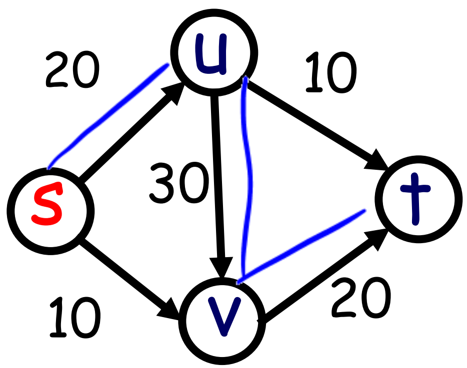


The max-flow here is 3.

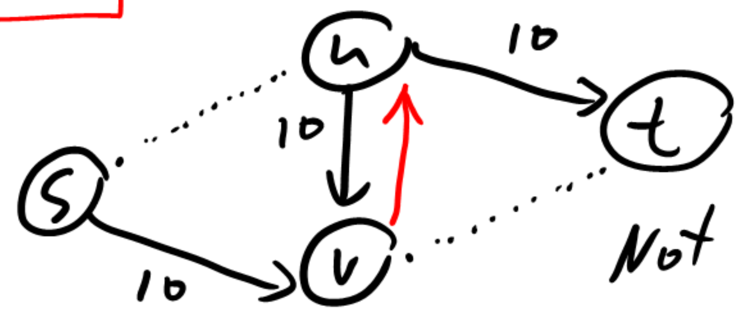
How can you see that the flow is really max?

Max-flow Problem: Given NF, find the max. flow from s to t.

Greedy Approach

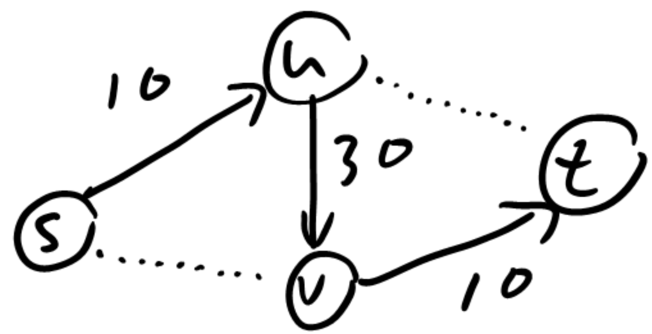


Push 20 via s-u-v-t



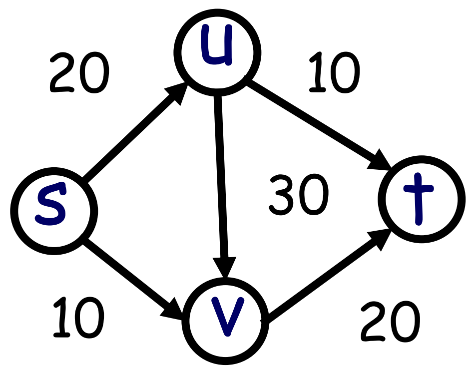
Not optimal!

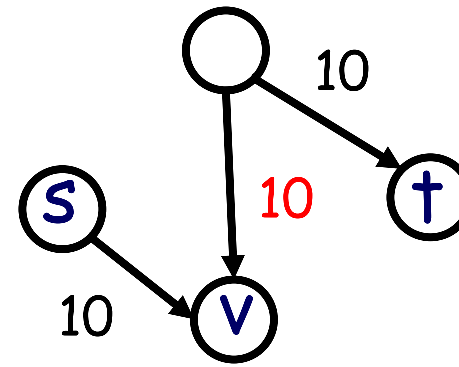
Push 10 via s-u-t
Push 10 via s-v-t
Push 10 via s-u-v-t
max-flow is 30



Canceling Flow

Push 20 via s-u-v-t





To increase a flow:

1) find unused capacity

2) find cancelable flow

Residual Graph G_f

$G = (V, E)$

$G_f = (V, E_f)$

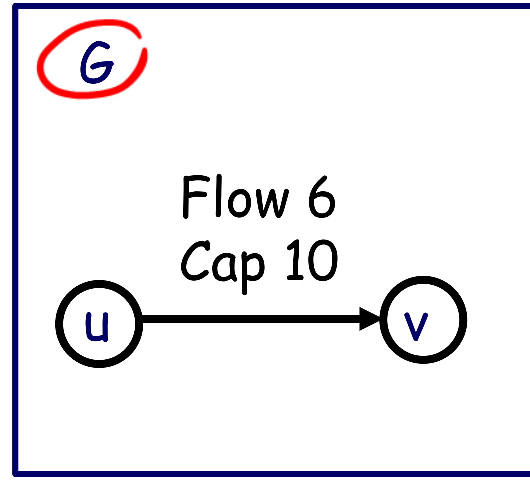
E_f consists of

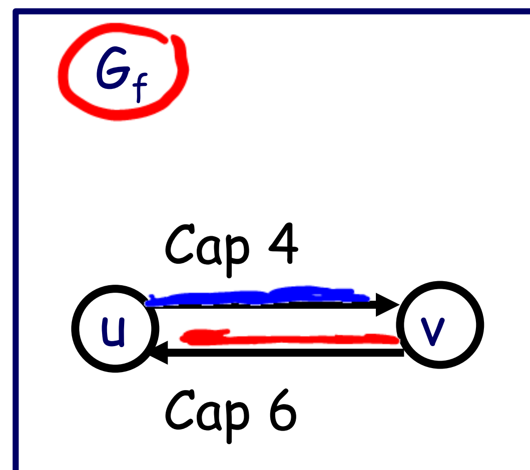
1) forward edges, $e \in E$

$$c_f(e) = c(e) - f(e) = 10 - 6 = 4$$

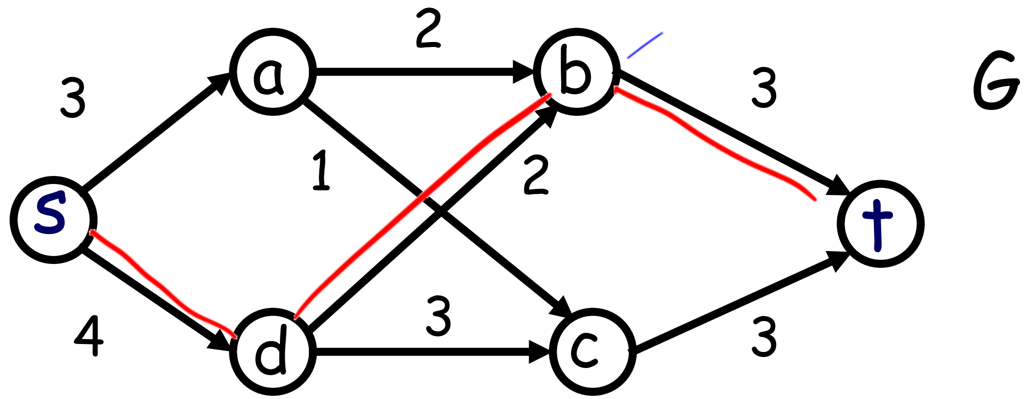
2) backward edges, $e \notin E$

$$c_f(e) = f(e)$$





Example: residual graph

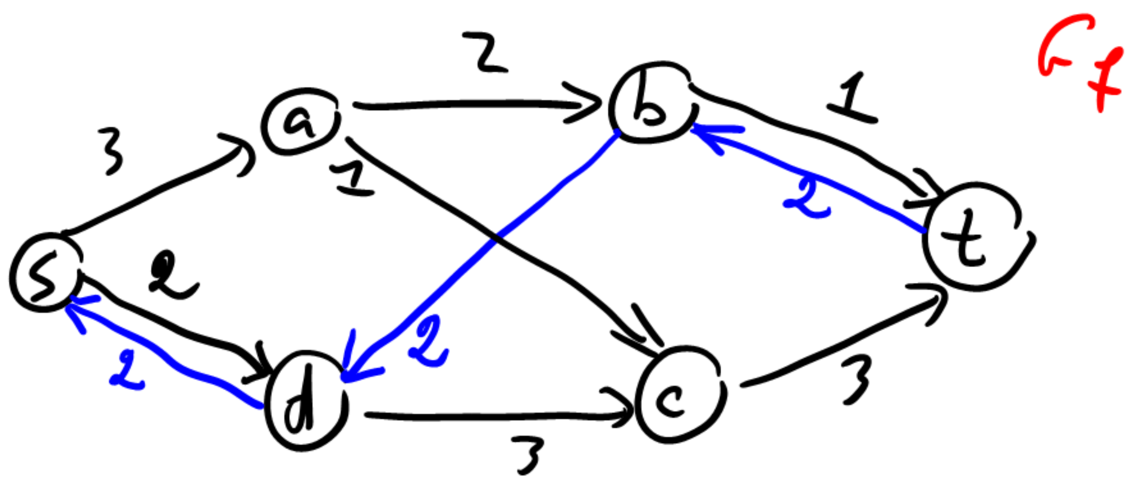


Push 2 along s-d-b-t and draw the residual graph

flow = 2

Path

s-a-b-d-c-t



Augmenting Path = Path in G_f

Let P be an $\boxed{\text{s–t path}}$ 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) along the path P.

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

$c_f(e) = c(e) - f(e)$

The Ford-Fulkerson Algorithm

<u>Algorithm</u>. 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

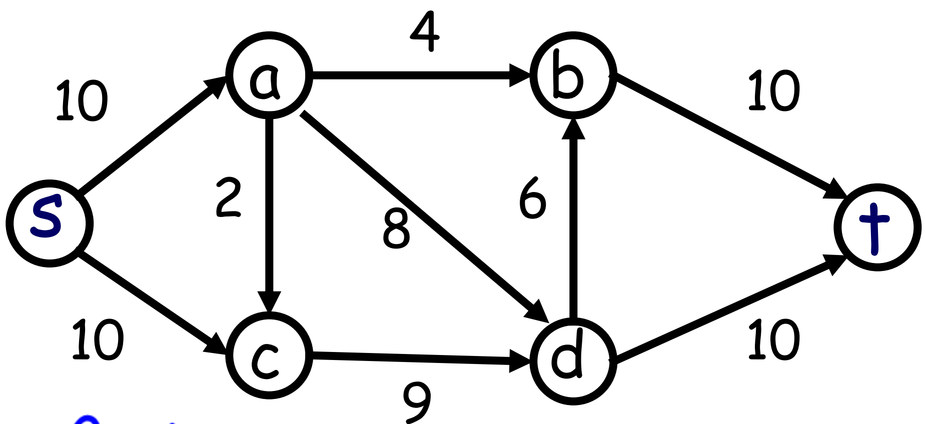
find bottleneck

augment the flow along this path

update the residual graph G_f

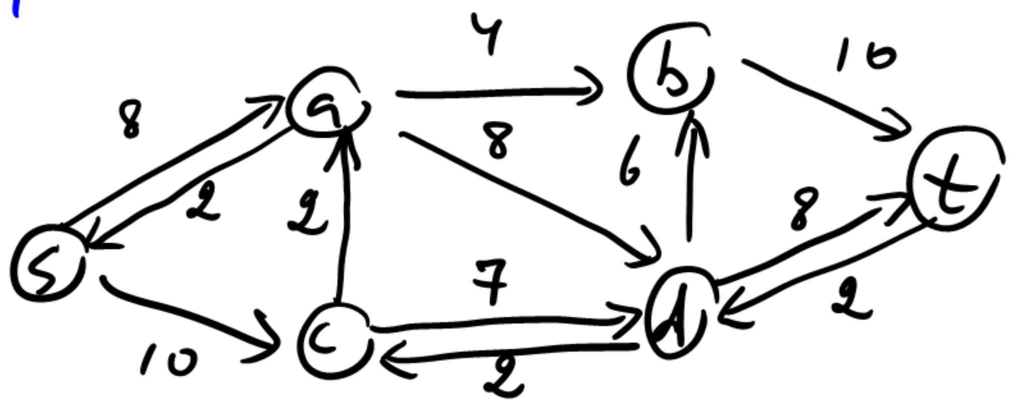
Example

$f = 0$
$G_f = G$



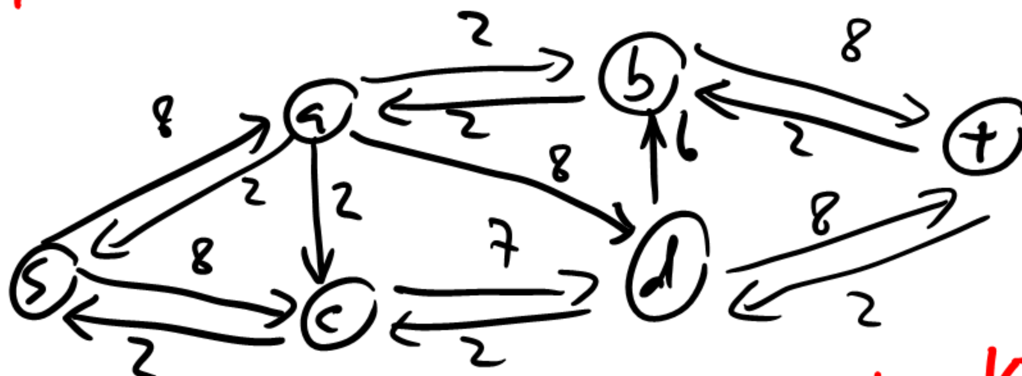
flow = 2

Path s-a-c-d-t, Push 2

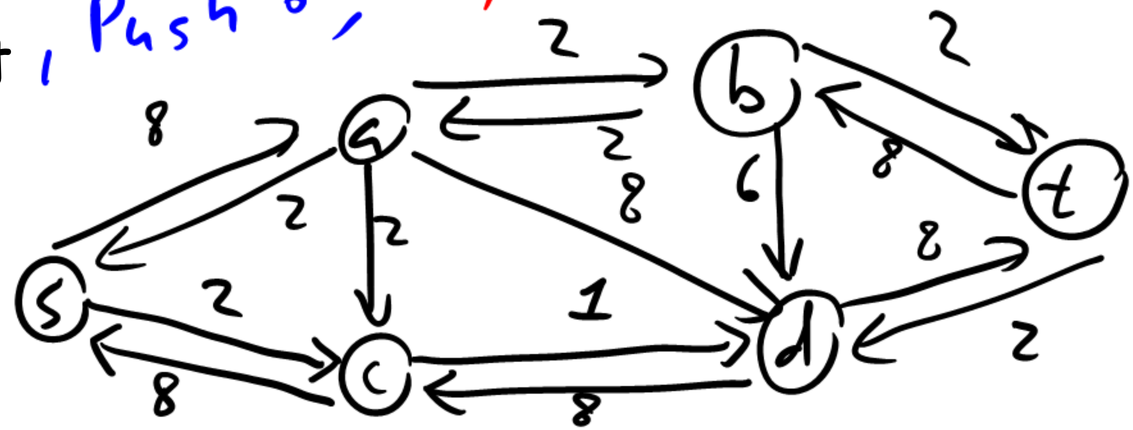


Example

Path s-c-a-b-t , Push 2 , flow = 2+2 = 4



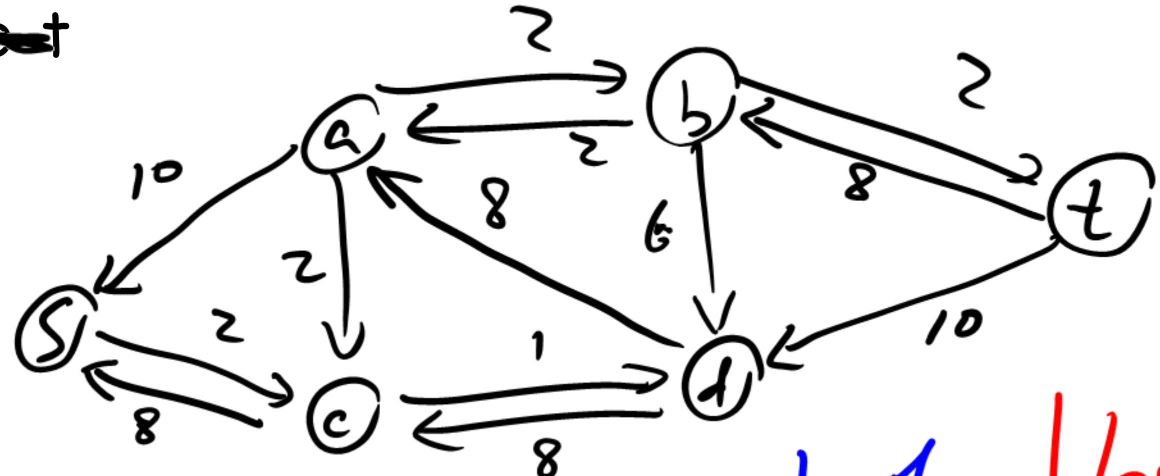
Path s-c-d-b-t , Push 6 , flow = 4+6 = 10



Example

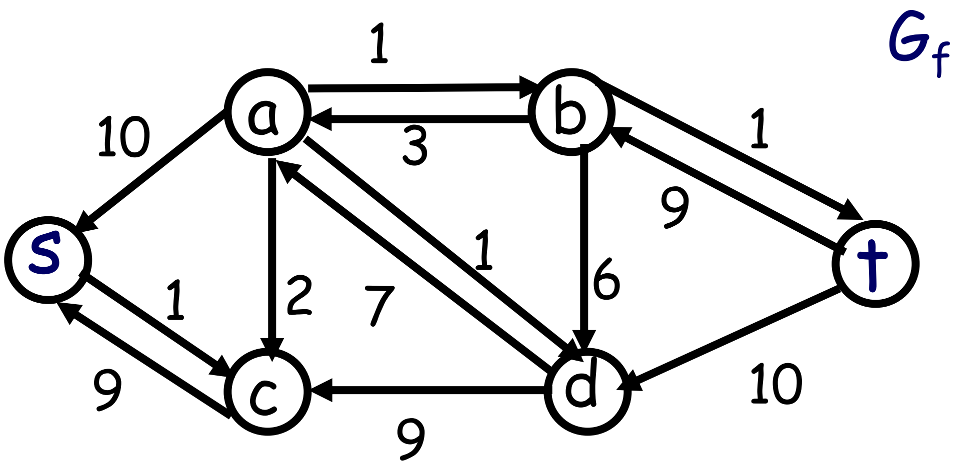
s-a-d-t Push 8, flow = 10+8 = 18

Path ~~s-c-d-a-b-t~~

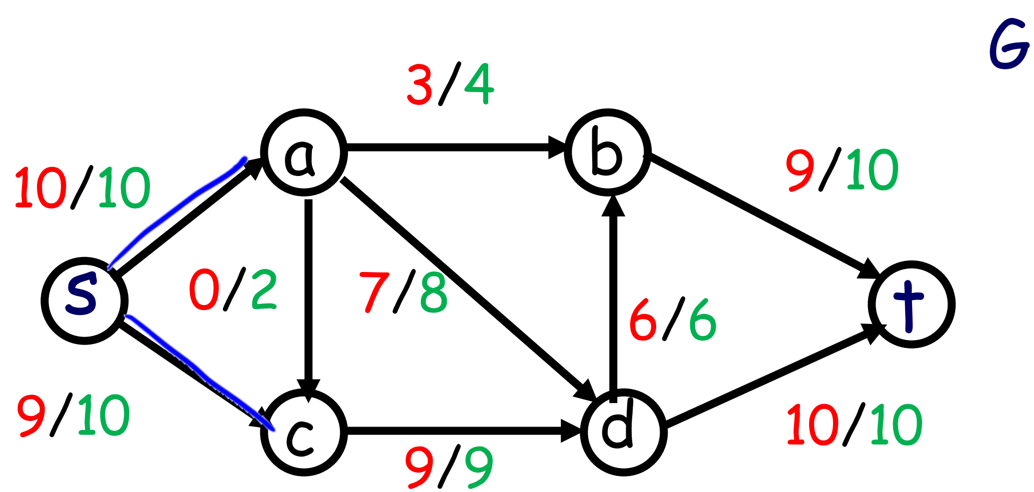


Path s-c-d-a-b-t, Push 1, flow = 19

G_f has no s-t path



In graph G edges are with flow/cap notation



Value of a flow

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

The Ford-Fulkerson Algorithm

Runtime Complexity

$c \in \mathbb{N}$, $f \in \mathbb{N}$

how do you find it?
DFS or BFS

<u>Algorithm</u>. Given (G, s, t, c)

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

while exists an augmenting path in G_f s-t

# of iterations in worst-case |f|

find <u>bottleneck</u> (smallest = 1)

<u>augment</u> the flow along this path

update the residual graph

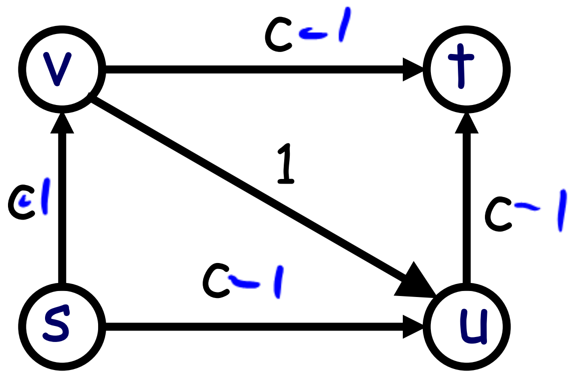
$O(\boxed{|f|} \cdot (V+E))$

number

Is it polynomial?
NO

The worst-case

$$\boxed{O(|f|\,(E+V))}$$



$c=10^9$

s-t path s-v-u-t
$f=1$

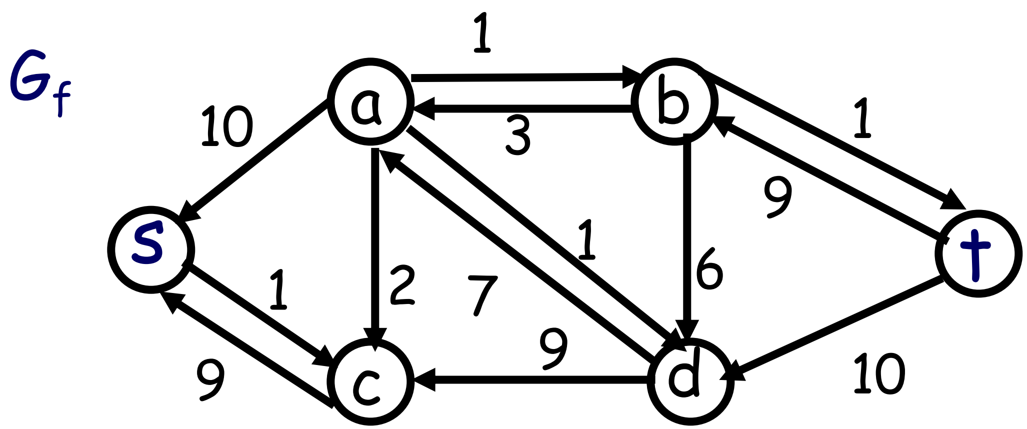
s-t path s-u-v-t
$f=2$

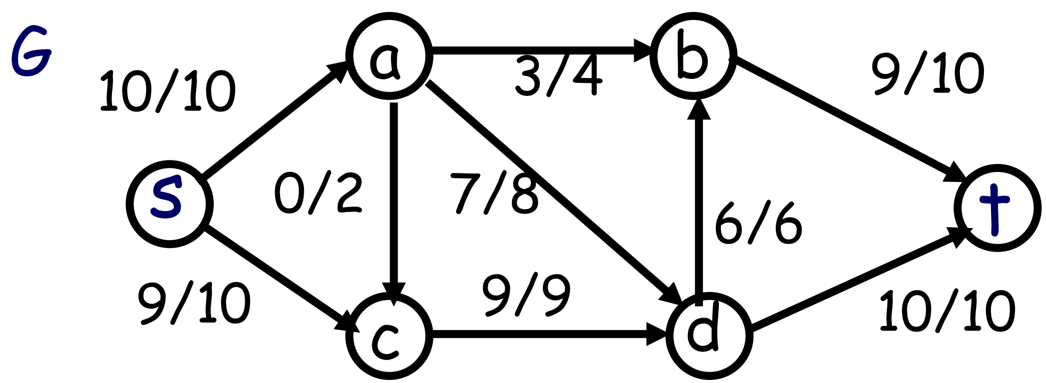
s-t path s-v-u-t
$f=3$

iterations $= 2 \cdot c = 2 \cdot 10^9$

Proof of Correctness

How do we know the flow is maximum?



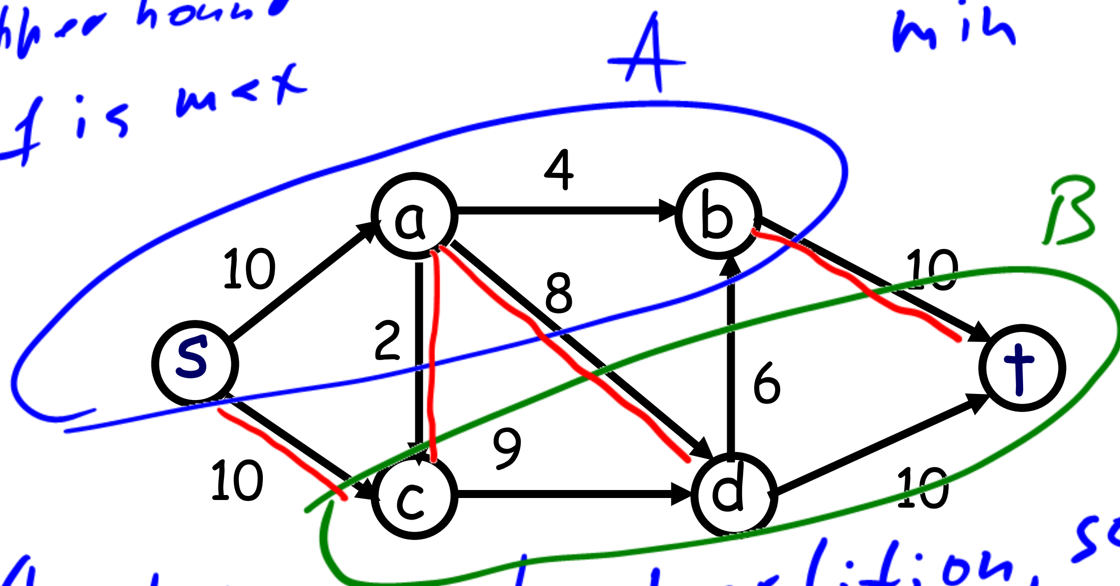


Cuts and Cut Capacity

$f \leq$ upper bound

if $f =$ upper bound then f is max

min



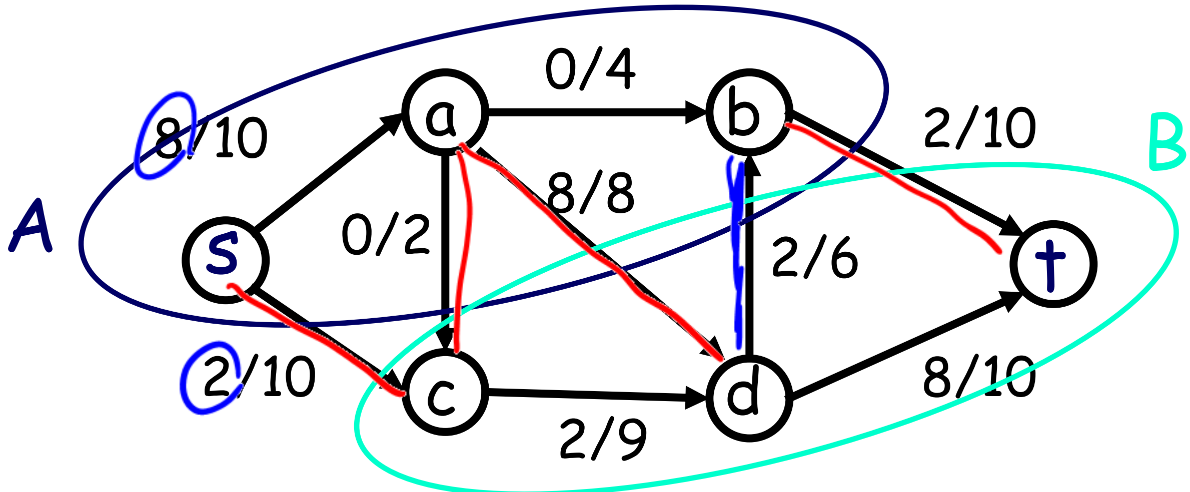
Def. A cut is a vertex partition, $s \in A$, $t \in B$

Def. $cap(A,B) = \sum_{\text{out of } A} c(e) = 10+2+8+10 = 30$

min-cut $\min_{A,B} cap(A,B)$

Cuts and Flows

Consider a graph with some flow and cut



The flow-out of A is $\underline{\quad}$ 2+0+8+2 = 12

The flow-in to A is $\underline{\quad}$ 2

The flow across (A,B) is $\underline{\quad}$ 12-2 = 10

What is a flow value |f| in this graph? 10

Lemma 1

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

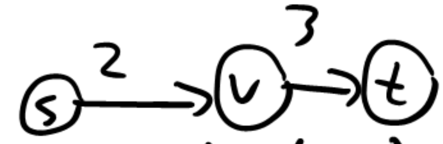
$$|f| = \sum_{v} f(s,v) = \sum_{u\in A, v\in B} f(u,v) - \sum_{u\in A, v\in B} f(v,u)$$

Proof.

$$|f| = \sum_{\text{out } s} f(e) = \sum_{\text{out } s} f(e) - \underbrace{\sum_{\text{to } s} f(e)}_{0} =$$

$$= \sum_{v\in A} \left[\sum_{\text{out } v} f(e) - \sum_{\text{to } v} f(e) \right]$$

$$= \sum_{\text{out } A} f(e) - \sum_{\text{to } A} f(e)$$



$2 = (2-0) + (2-2)$

Lemma 2

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

$$|f| \leq cap(A,B)$$

flow capacity

duality

$$\max_f |f| \leq \min_{(A,B)} cap(A,B)$$

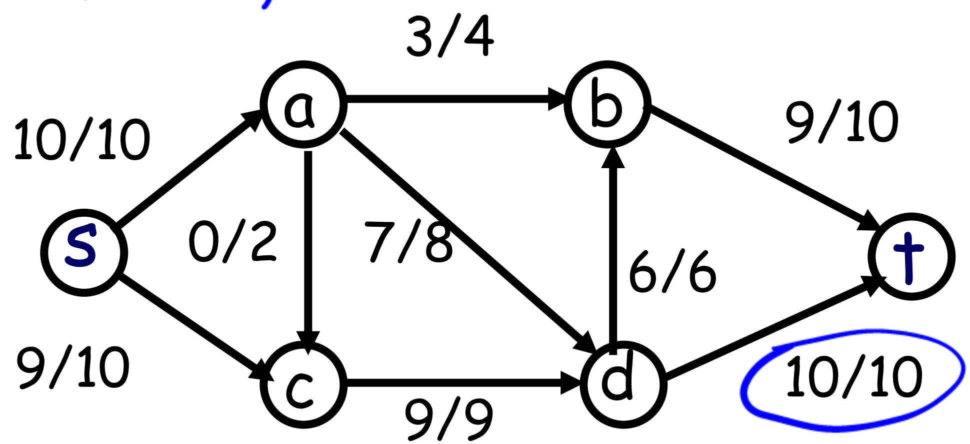
Proof: Use lemma 1

$$|f| = \sum_{\text{out } A} f(e) - \cancel{\sum_{\text{to } A} f(e)} \leq \sum_{\text{out } A} f(e) \leq \sum_{\text{out } A} c(e)$$

$$\sum_{\text{out } A} c(e) = cap(A,B)$$

Max-flow Theorem

Theorem. *The Ford-Fulkerson algorithm outputs the maximum flow.* = integer



$c(e) > 0$

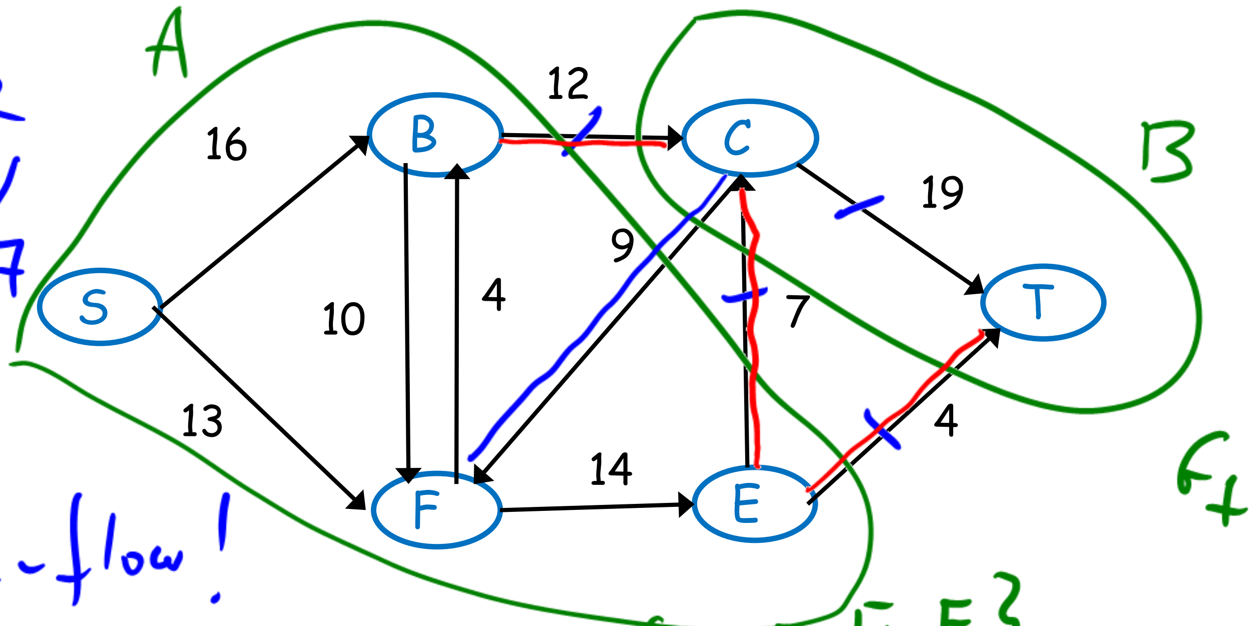
$c(e) \in \mathbb{N}$

$10. - 10. = 0, 0\ldots$

Discussion Problem 1

Run the Ford-Fulkerson algorithm on the following network:

S-B-C-T: 12
S-F-E-T: 4
S-F-E-C-T: 7



Find max-flow!
23

$A = \{S, B, E, F\}$
$B = \{T, C\}$

$A = \{S, B, E, F, C\}$
$B = \{T\}$

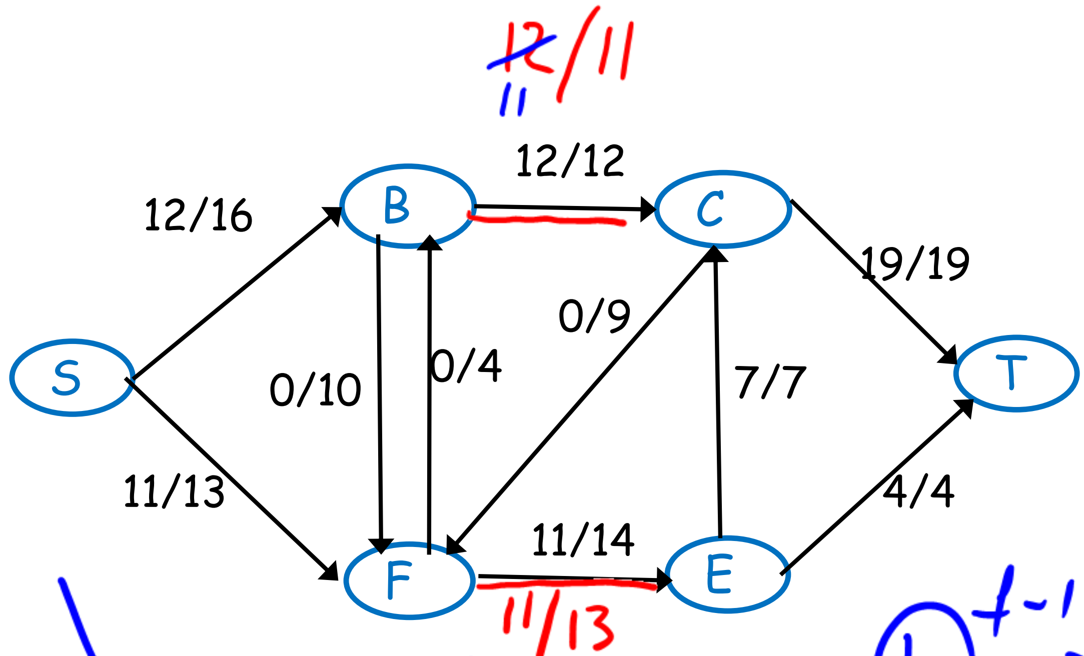
How do you find a min-cut ?

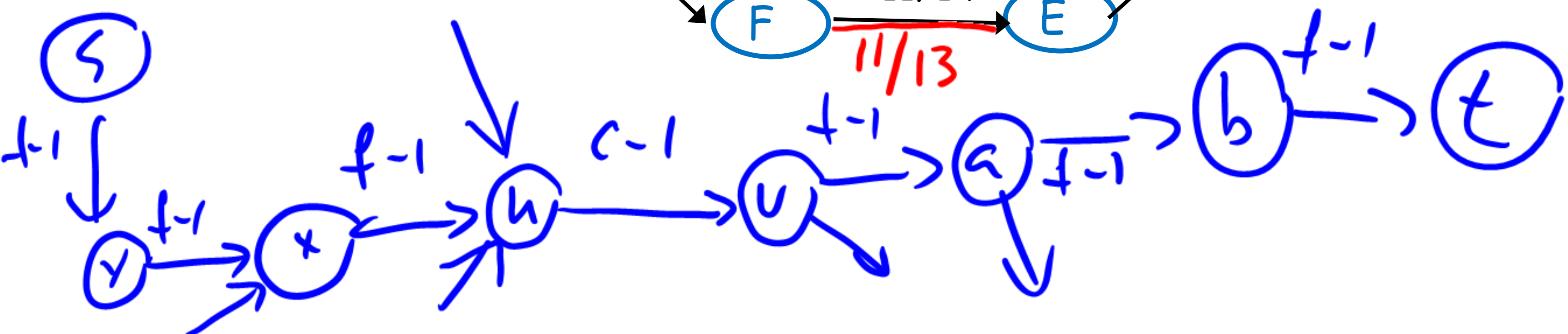
Is a min-cut unique? No

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.

?





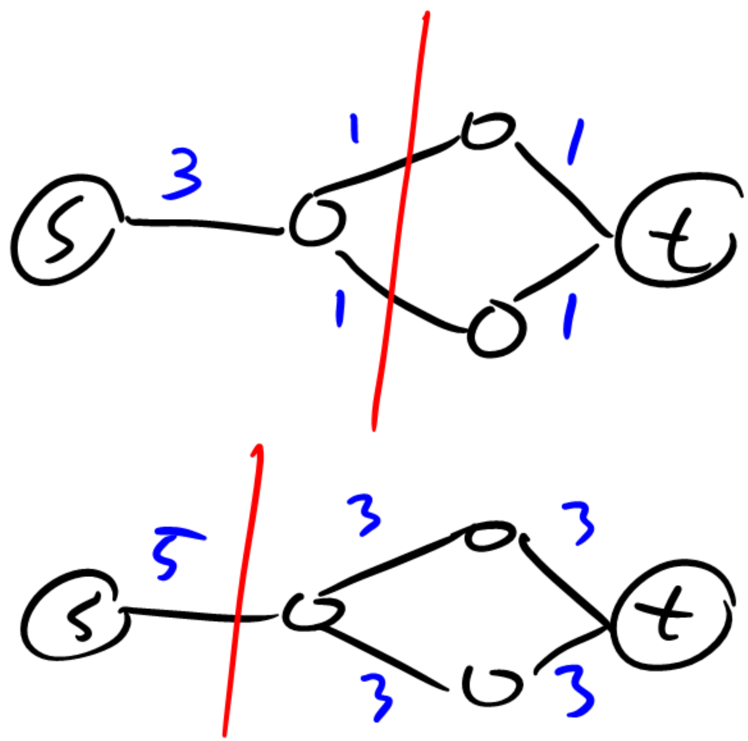
find s-t path flow: f

a) ∃, increase

b) ∄, f-1 is max

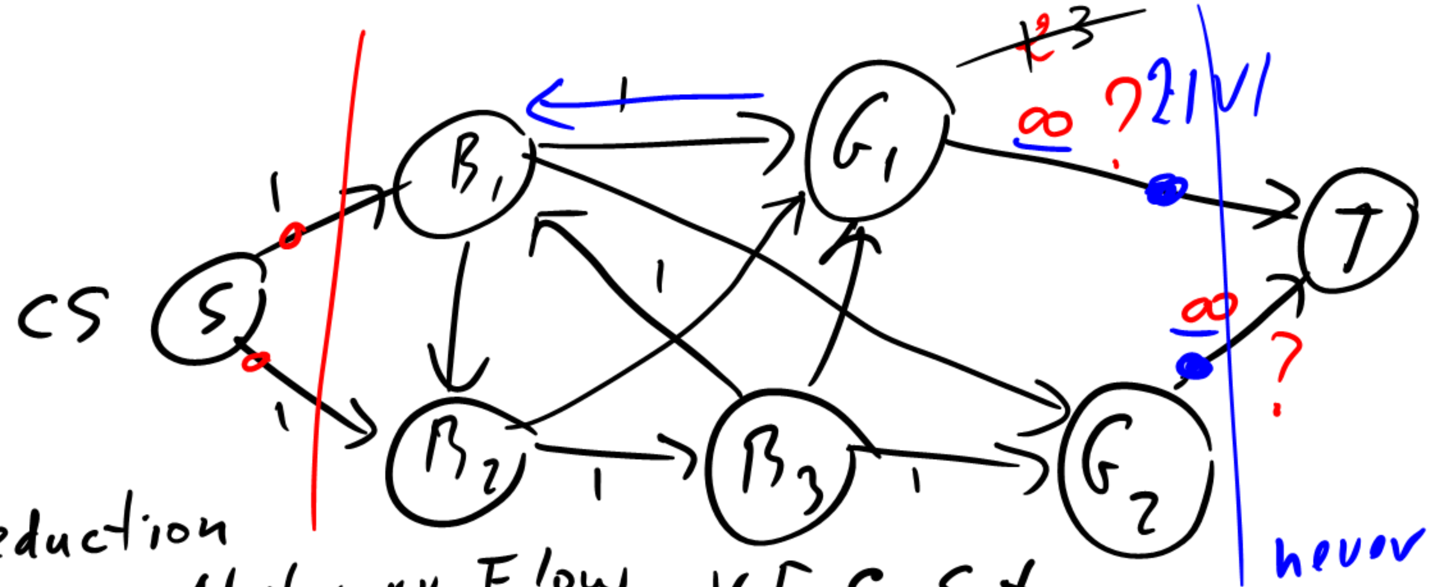
Discussion Problem 3

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.



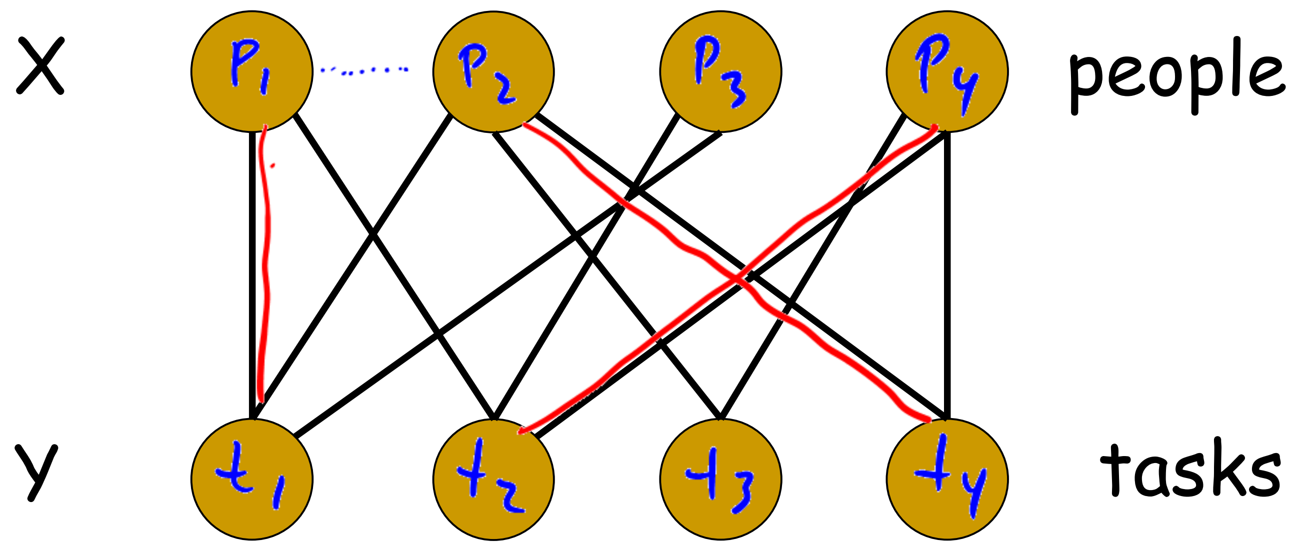
By reduction

1) create Network Flow, V, E, c, S, t

2) run FF

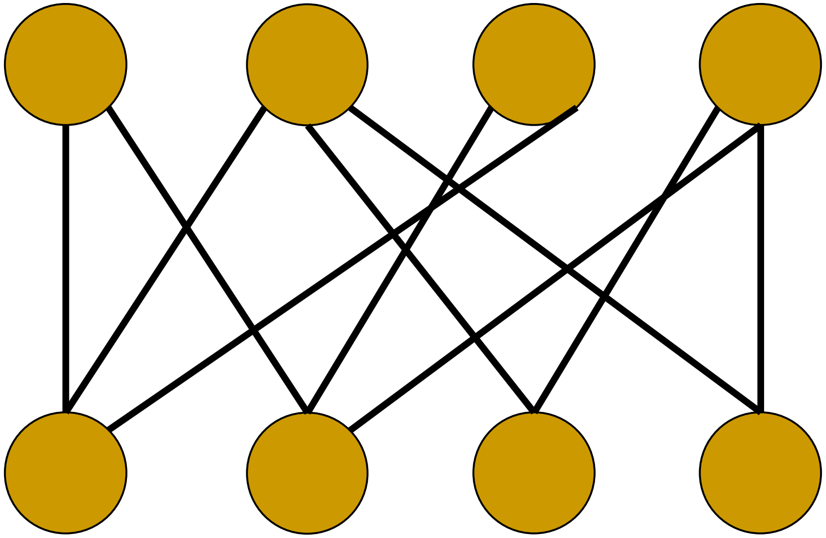
Placement -? min-cut

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

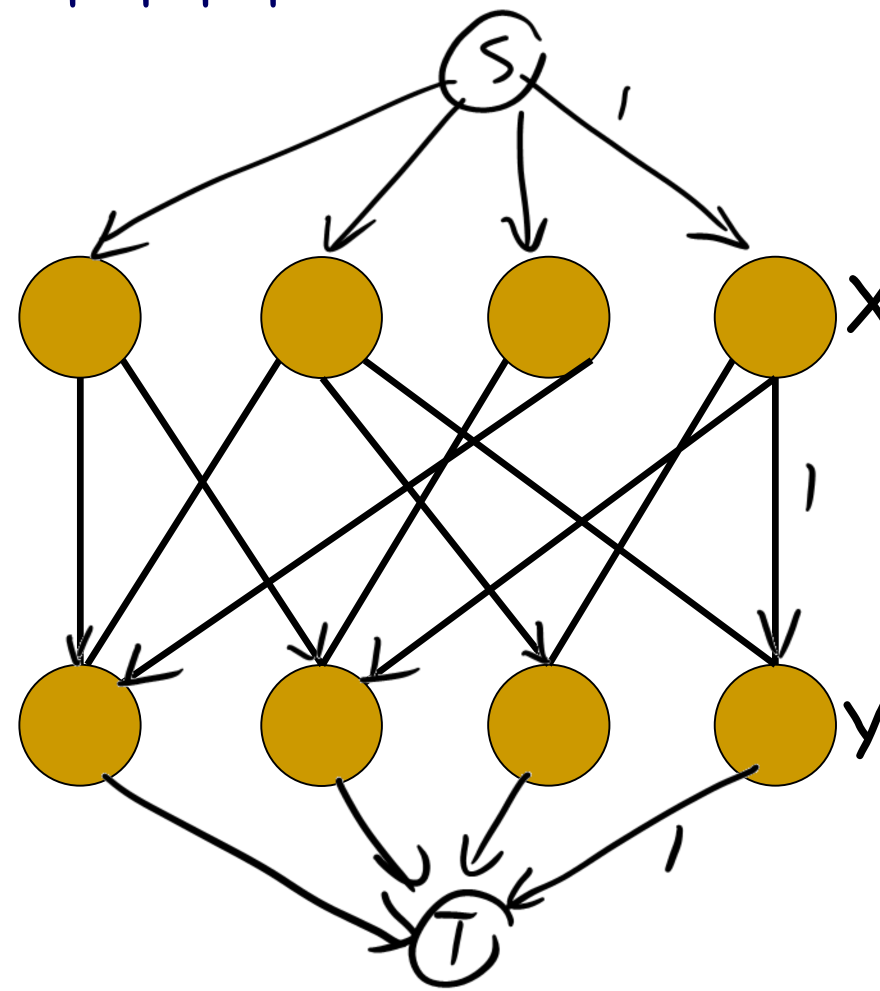
$$\text{Input}(BM) \xrightarrow[V,E,s,t,c]{NF} \text{Input}(NF)$$

FF

$$\text{Output}(BM) \xleftarrow[\text{claim.}]{} \text{Output}(NF)$$

Reducing Bipartite Matching to Network Flow

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



Claim:

max-matching = max-flow

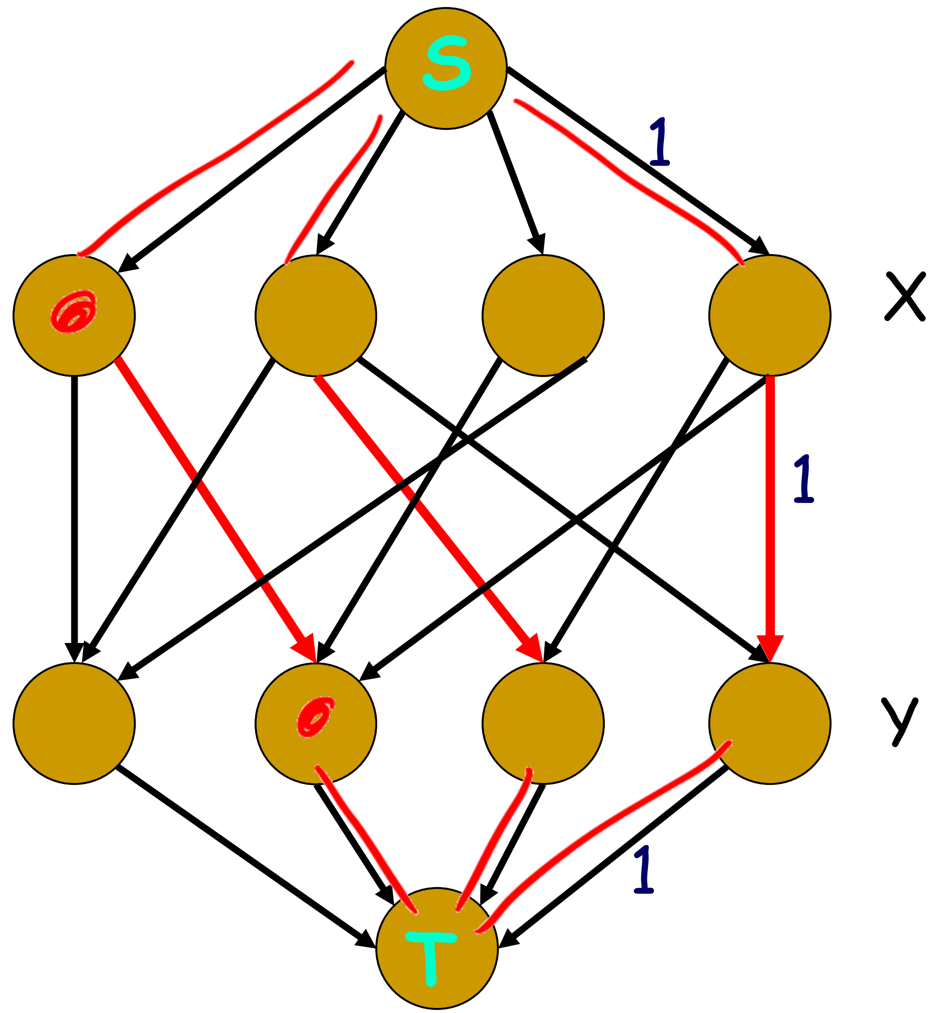
$\Rightarrow$)

$\Leftarrow$)

Max matching = Max flow

$\Rightarrow$ $\exists$ matching of size K, then ~~max~~-flow is K

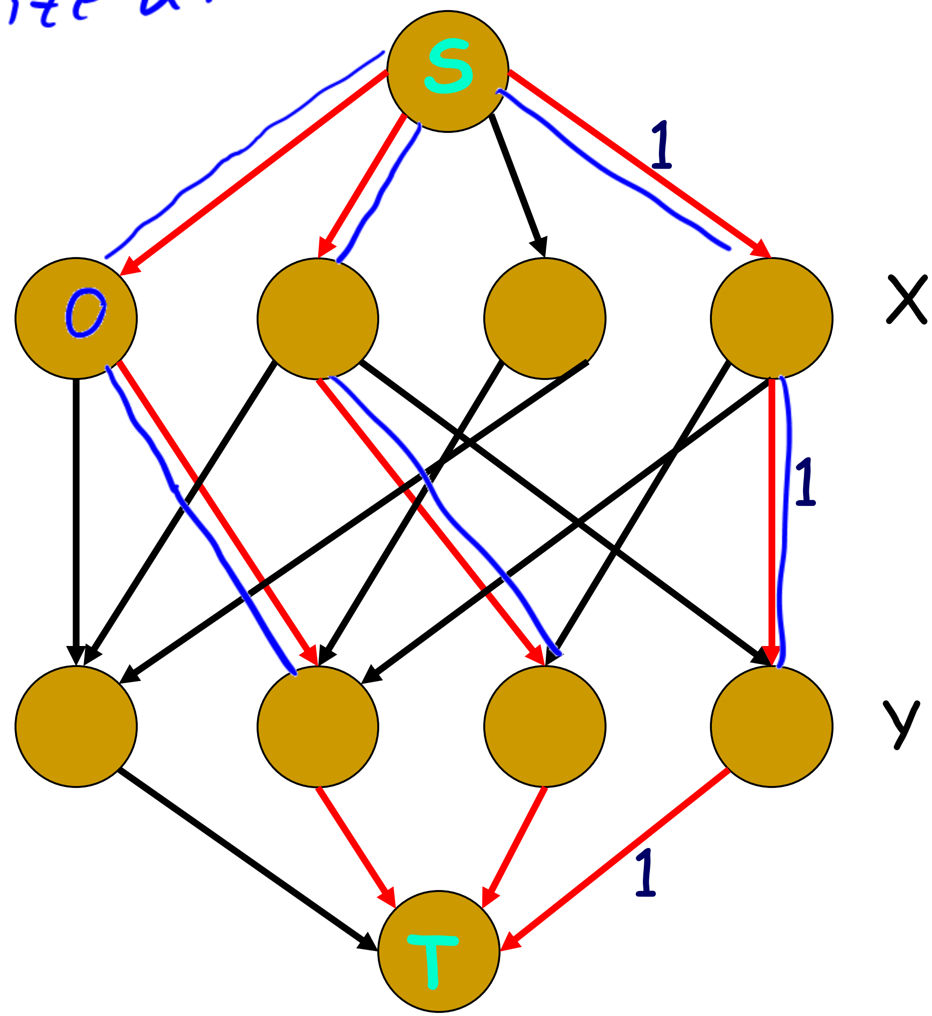
Proof.



Max matching = Max flow

$\Leftarrow$) $\exists$ flow of value k,
then $\exists$ matching of size k.

Proof.



Runtime Complexity

Given bipartite $G = (X, Y, E)$. , $\underline{|X|=|Y|=V.}$

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

$N=(V_1, E_1)$, $V_1 = 2V+2$, $E_1 = E+2\cdot V$, $|f|=V$

$O(|f|(V_1+E_1)) = O(V\cdot(2V+2+E+2V))$

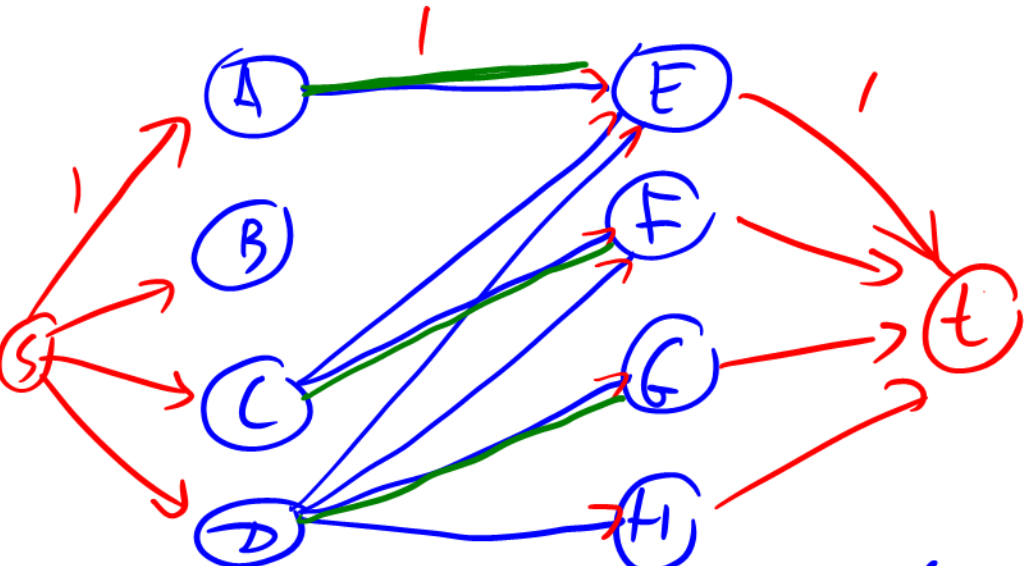
$= O(V\cdot E)$

is it polynomial?

Yes

Discussion Problem 5

We're asked to help the captain of the USC tennis team to arrange a series of matches against UCLA's team. Both teams have n players; the tennis rating of the i-th member of USC's team is a_i and the tennis rating for the k-th member of UCLA's team is b_k. We would like to set up a competition in which each person plays one match against a player from the opposite school. Our goal is to make as many matches as possible in which the USC player has a higher tennis rating than his or her opponent. Give an algorithm to decide which matches to arrange to achieve this objective.



Player	Rating	Team
A	10	Trojans
B	5	Trojans
C	15	Trojans
D	20	Trojans
E	7	Bruins
F	14	Bruins
G	16	Bruins
H	19	Bruins

$\boxed{\text{bears}}$

Input(Tennis) $\rightarrow$ Input(BM) $\rightarrow$ Input(NF)

FF

Output(Tennis) $\leftarrow$ Output(BM) $\leftarrow$ Output(NF)

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

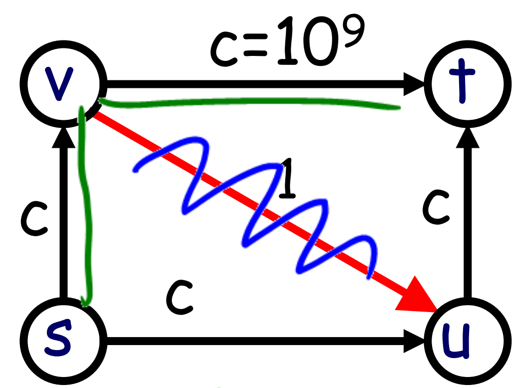
Edmond-Karp

FF + BFS (to find s-t path)

$O(V \cdot E^2)$, polynomial

(No proof)

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



DFS:
s-v-u-t
len = 3

BFS:
s-v-t

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 (BFS)
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

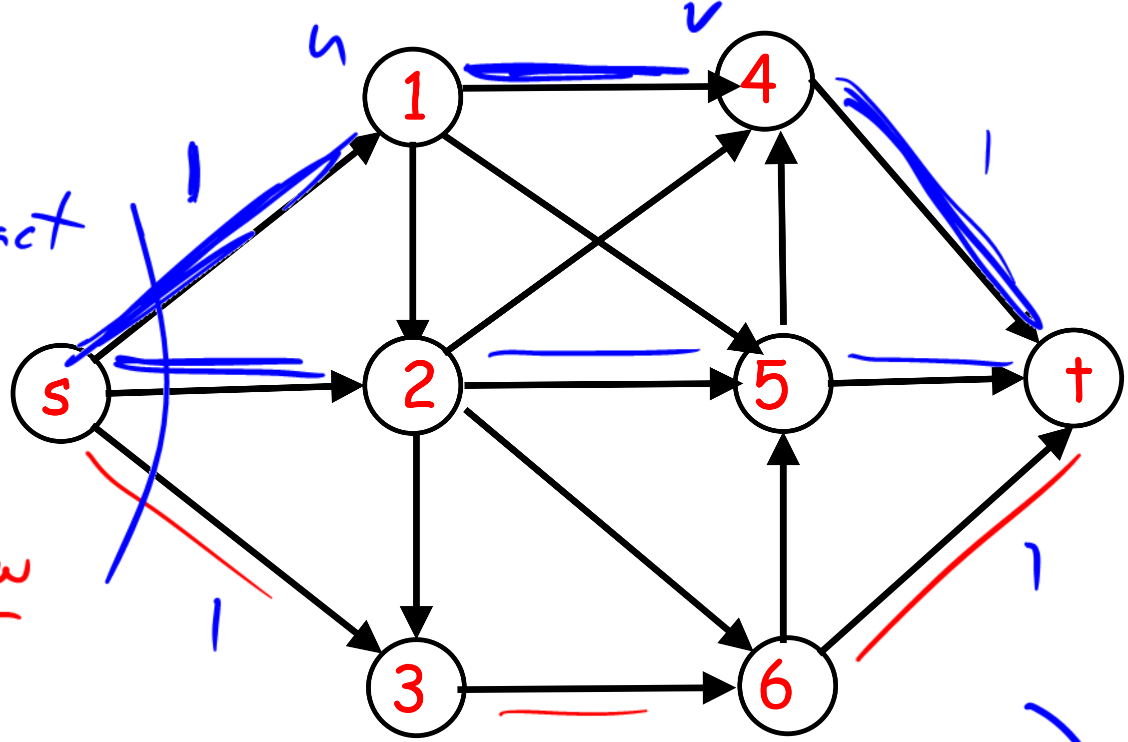
n = V, m = E,
U = |f|

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 practical
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) theoretical

Discussion Problem 6

Professor Jones has determined that x priceless artifacts are located in a labyrinth. The labyrinth can be thought of as a graph, with each edge representing a path and each node an intersection of paths. All of the artifacts are in the same treasure room, which is located at one of the intersections. However, the artifacts are extremely burdensome, so Jones can only carry one artifact at a time. There is only one entrance to the labyrinth, which is also a node in the graph. The entrance serves as the only exit as well. All the paths are protected by human-eating vines, which will be woken up after someone passes the path, so Jones can only go through each path once. Give an algorithm that determines how many artifacts Jones can obtain and how he can do it.

1 artifact



artifacts

$\frac{\text{max-flow}}{2}$

$$\#\text{artifacts} = (\text{edge-disjoint paths})/2$$

Claim. max-flow = max # of disjoint paths.

$\Rightarrow$) $\exists$ flow of value K.

Consider (s,u) edge.

By conservation law, $\exists$ edge (u,v), $f=1$

Continue until reach T.

Repeat.

$\Leftarrow$) $\exists$ K edge-disjoint paths

Since capacity is $\underline{1}$, each edge can be used only once.

$G=(V,E)$

Runtime Complexity.

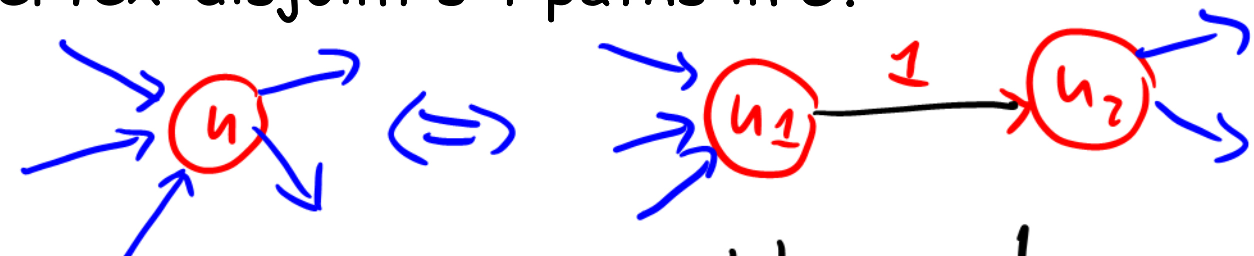
FF: $O(|f|\cdot E)$, $\boxed{|f|=V}$

$O(V\cdot E)$

Discussion Problem 7

We say that two paths are vertex-disjoint if they do not share any vertices (except s and t).

Given a directed graph G = (V, E) with two distinguished nodes s, t. Design an algorithm to find the maximum number of vertex-disjoint s-t paths in G.



Reduced vertex-disjoint problem to edge-disjoint problem

Claim. # VD paths = # ED paths

Discussion Problem 8

There are n students in a class. We want to choose a subset of k students as a committee. There has to be m_1 number of freshmen, m_2 number of sophomores, m_3 number of juniors, and m_4 number of seniors in the committee. Each student is from one of k departments, where $k = m_1 + m_2 + m_3 + m_4$. Exactly one student from each department has to be chosen for the committee. We are given a list of students, their home departments, and their class (freshman, sophomore, junior, senior). Describe an efficient algorithm based on network flow techniques to select who should be on the committee such that the above constraints are all satisfied.