Analysis of Algorithms

V. Adamchik

Lecture 8

Fall 2023

CSCI 570

University of Southern California

Network Flow

Reading: chapter 7.1 - 7.4

The Network Flow Problem

Solve by reduction

Our fourth major algorithm design technique

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

input $\longrightarrow$ input(NF)

solve(NF)

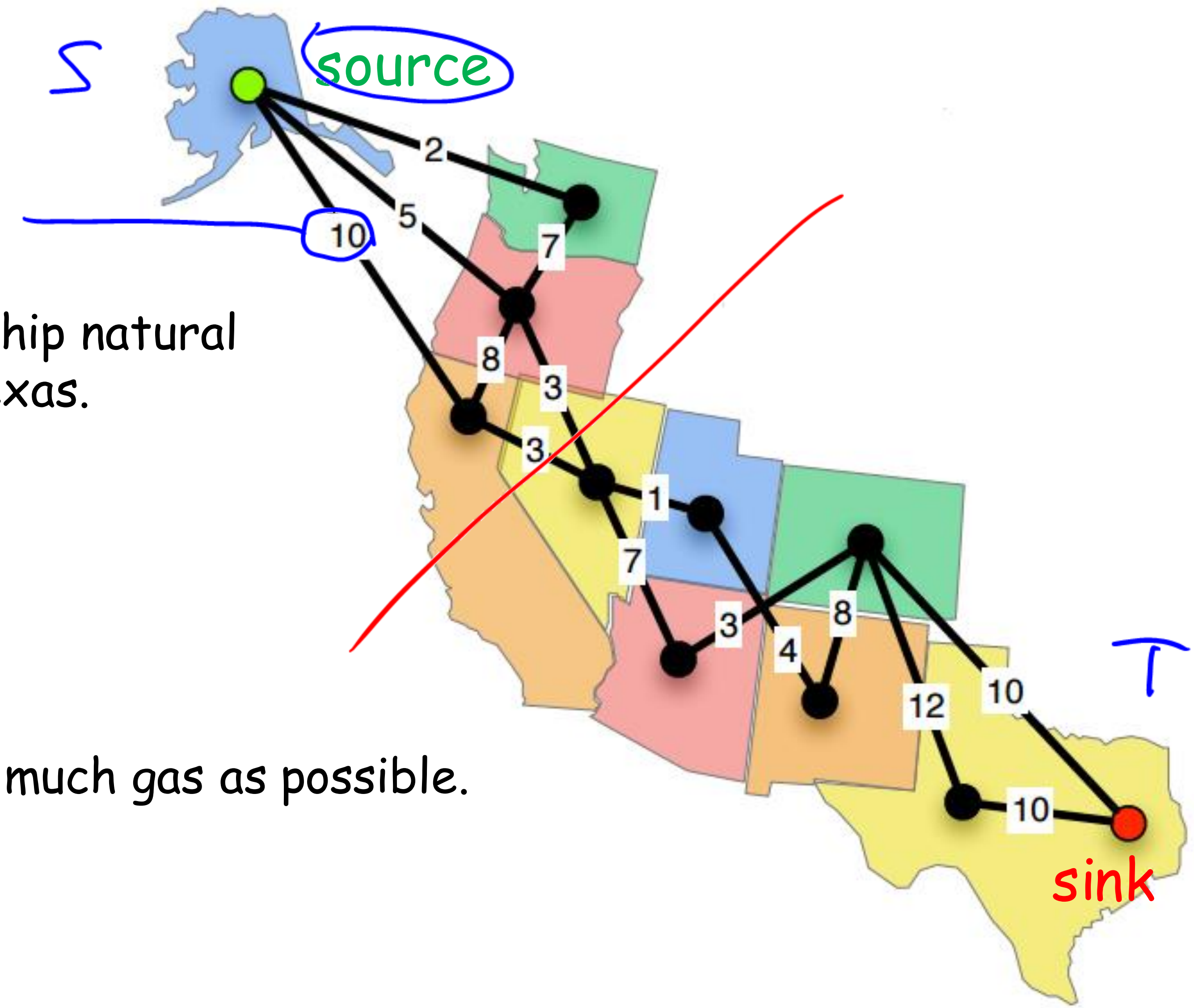
output $\longleftarrow$ output

Plan:

The Ford-Fulkerson algorithm

Max-Flow Min-Cut Theorem

The Flow Problem



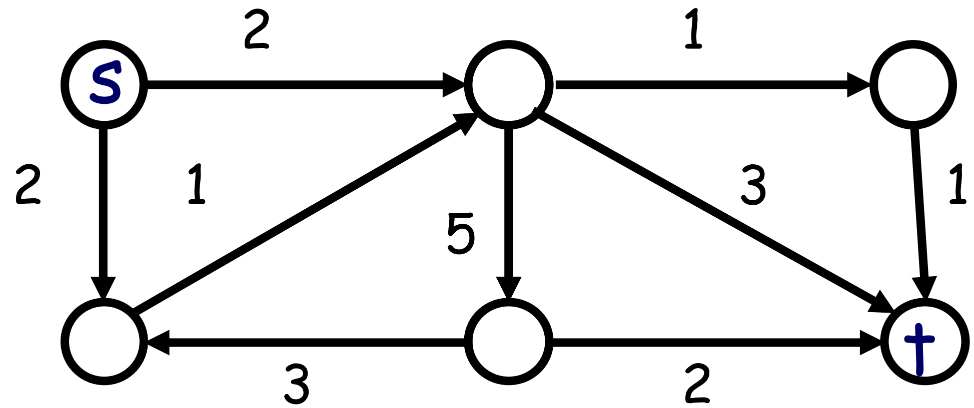
Suppose you want to ship natural gas from Alaska to Texas.

Pipes have capacities.

The goal is to send as much gas as possible.

How can you do it?

The Max-Flow Problem



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

$c > 0$

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

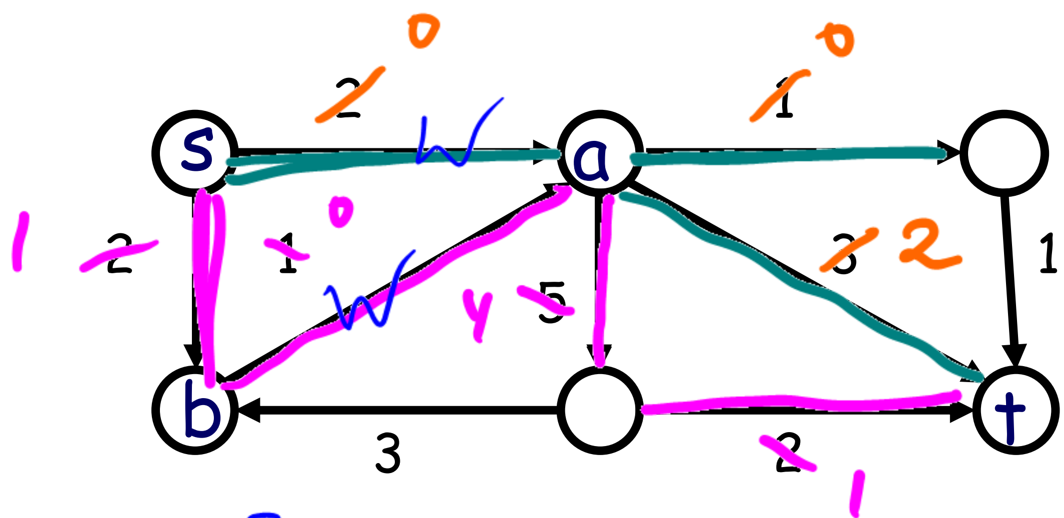
we define a *flow* as a function $f\colon E \to \mathbb{R}^+$ that assigns nonnegative real values to the edges of G and satisfies two axioms:

1. Capacity constraint:
2. Conservation constraint:

<u>Def.</u> a flow $f(e): e \to \mathbb{R}^+$

1. $0 \le f(e) \le c(e)$
2. $\sum_{\text{into } v} f(e) = \sum_{\text{out } v} f(e), \quad v \notin \{s, t\}$

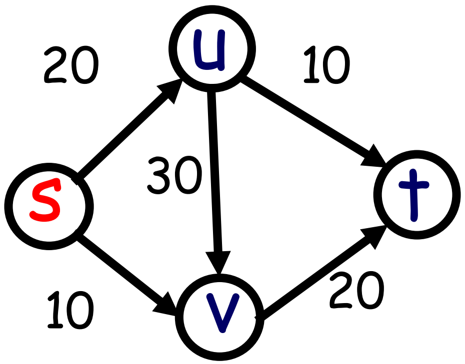
The MAX Flow Problem

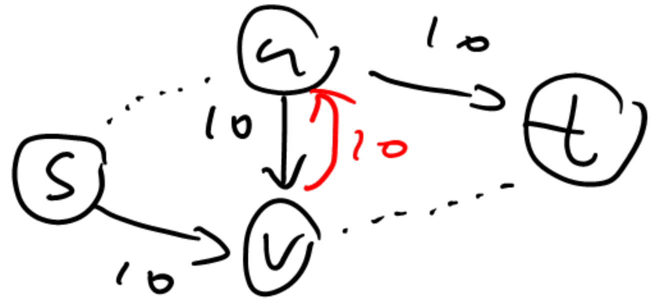


The max-flow here is 3.

How can you see that the flow is really max? ← ??

Greedy Approach: push the max





Push 20 via s - u - v - t

Another approach?

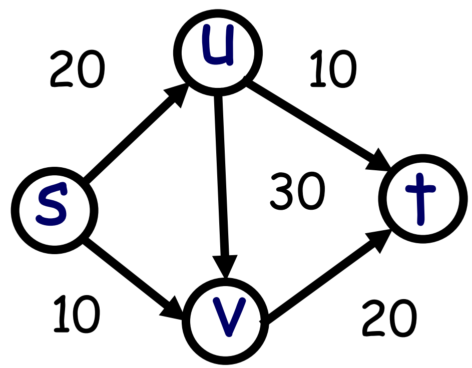
① push 10 via s-u-t

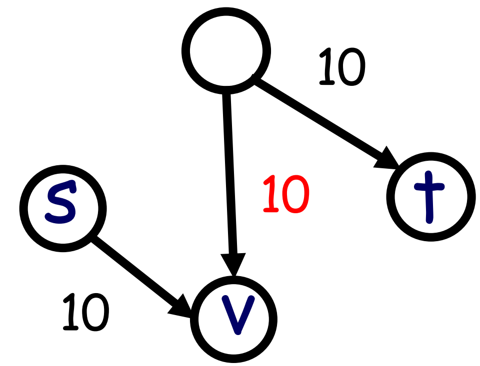
② push 10 via s-v-t

③ push 10 via s-u-v-t

Canceling Flow

Push 20 via s-u-v-t





modify graph by adding new edges

Residual Graph G_f
Residual Capacity c_f

$G = (V, E)$

$G_f = (V, E_f)$

E_f consists of

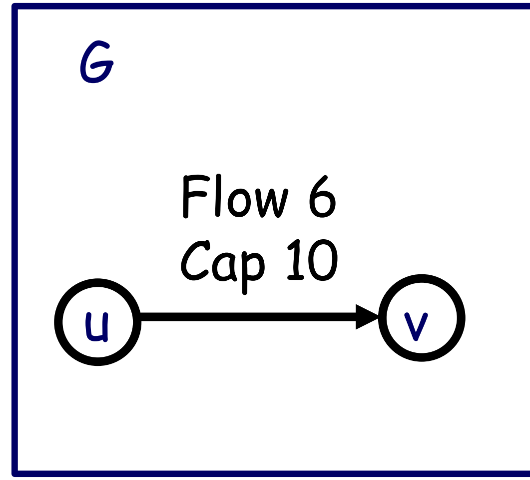
① forward edges, $e \in E$

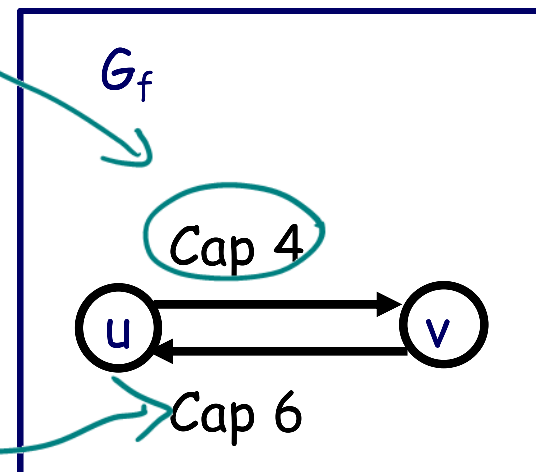
$c_f(e) = c(e) - f(e)$

② backward edges, $e \notin E$

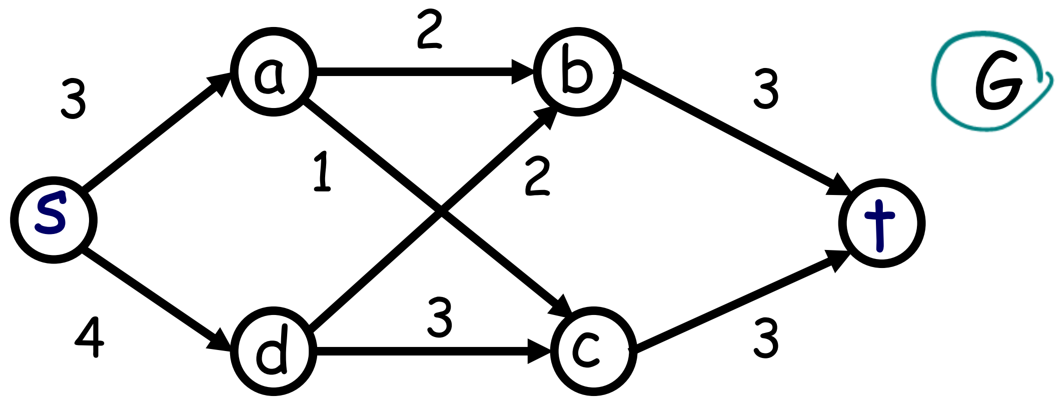
$c_f(e) = f(e) \geq 0$

$2.00001 - 1.9999 \neq 0$

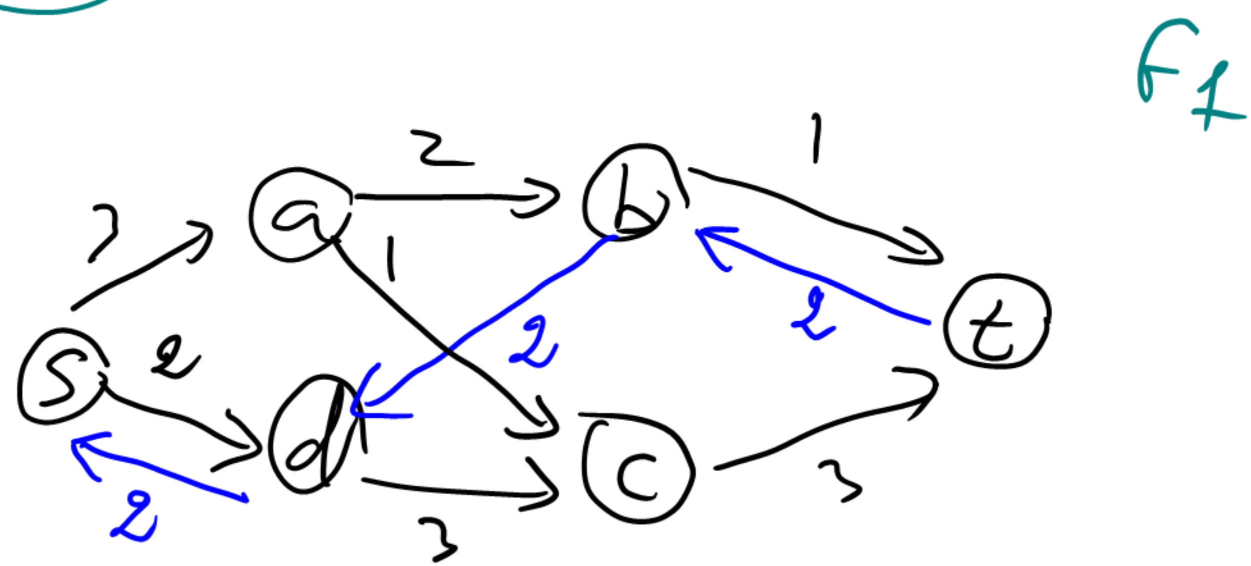




Example: residual graph



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

FF

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

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

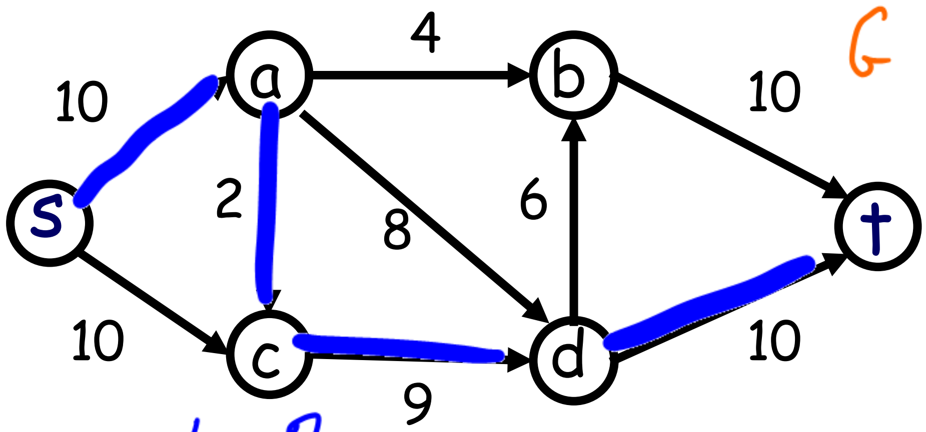
while exists an augmenting path in G_f

traversal find bottleneck

augment the flow along this path

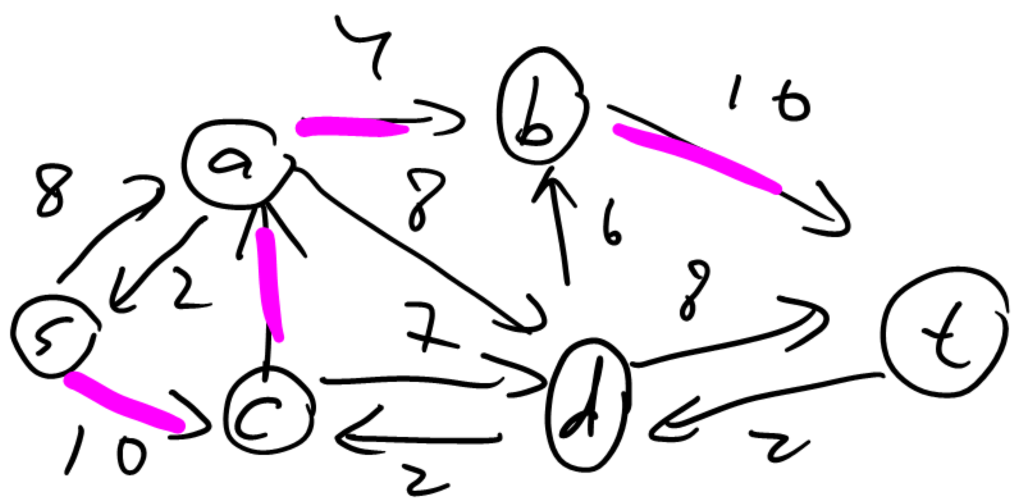
update the residual graph G_f

Example



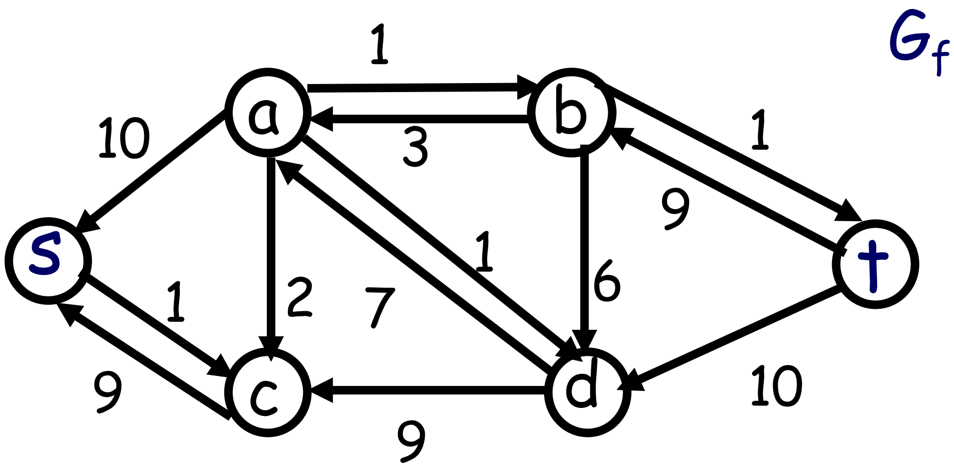
Path s-a-c-d-t , push 2

$G_f = G$

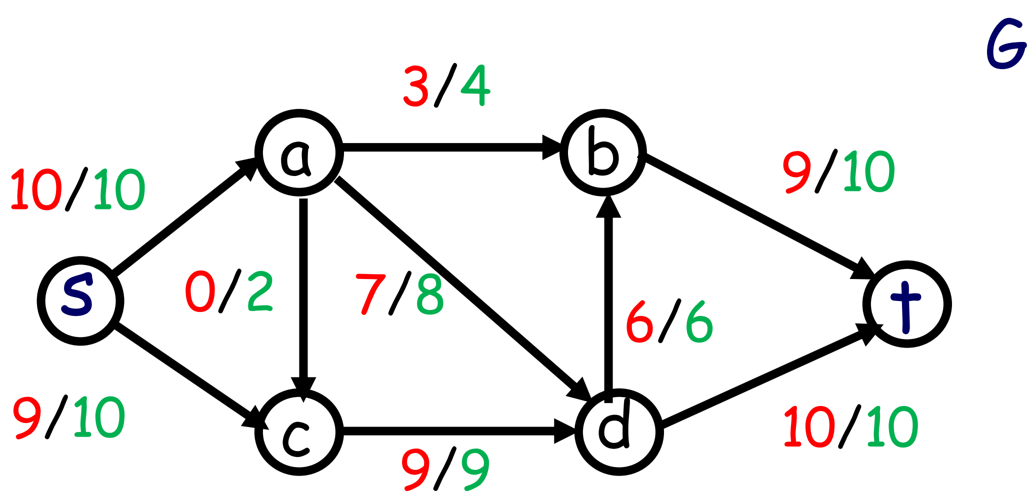


Example

Path s-c-a-b-t , push 2



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. → $O(V+E)$

while exists an augmenting path in G_f

steps find bottleneck → $O(V)$

augment the flow along this path $O(V)$

update the residual graph $O(V)$

$\# \text{ of steps} \le |f| = \sum_{\text{out } S} f(e)$

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

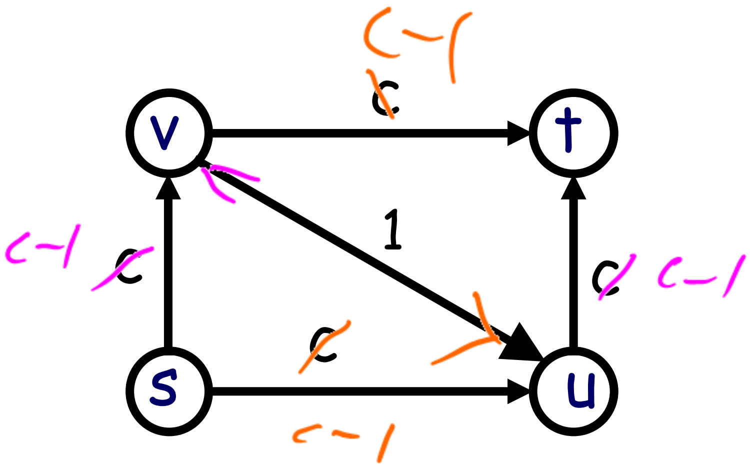
is it polynomial?

No

The worst-case

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

$c=10^9$



s-v-u-t, f = 1

s-u-v-t, f = 2

continue

# of iterations = 2 × C

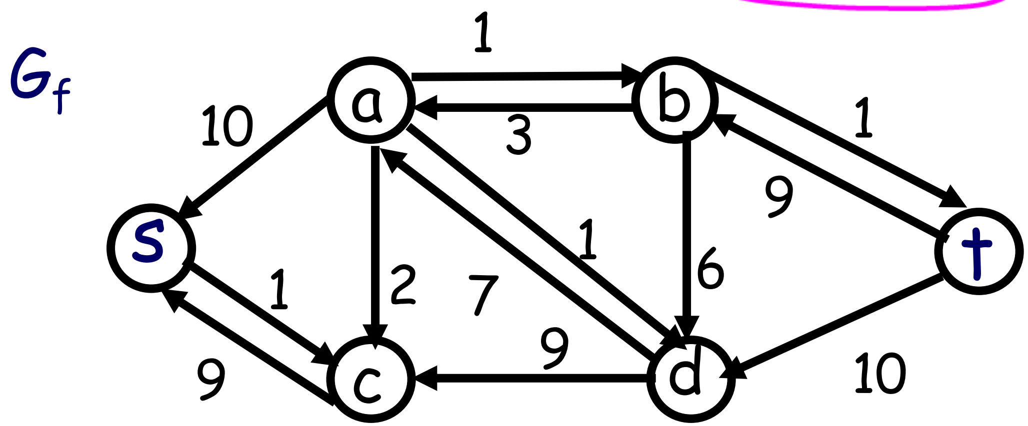
Break.

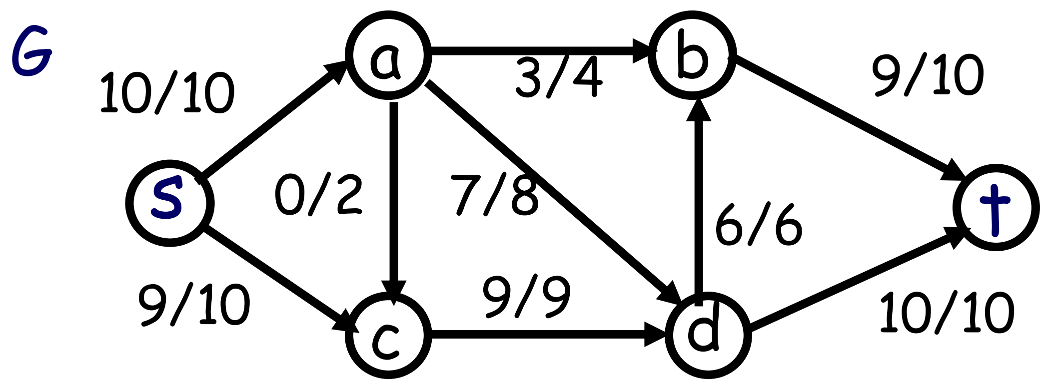
Proof of Correctness

$c \in \mathbb{N}^+$

How do we know the algorithm terminate

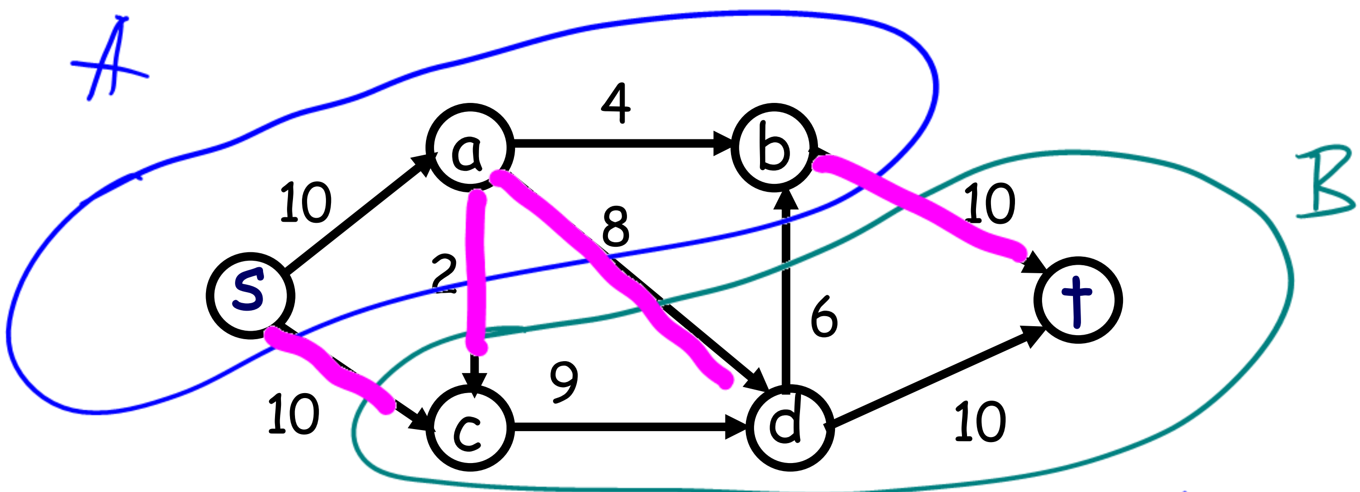
How do we know the flow is maximum? ?





Duality

Cuts and Cut Capacity



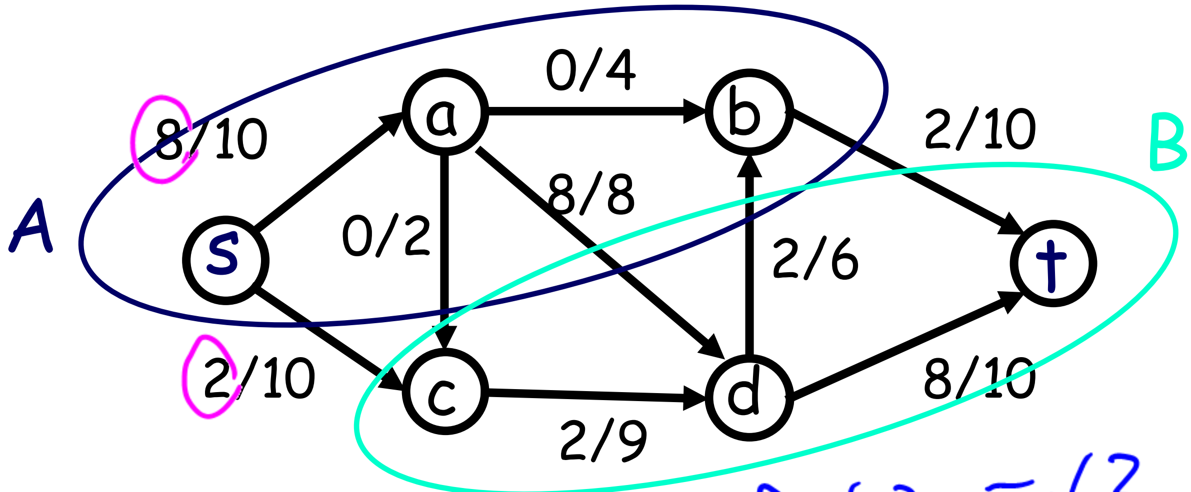
$s \in A$, $t \in B$

<u>Def 1</u> a cut is a vertex partition,

<u>Def.2</u> $cap(A, B) = \sum_{out\ A} c(e) = 30$

Cuts and Flows

Consider a graph with some flow and cut



The flow-out of A is ____ 2 + 0 + 8 + 2 = 12

The flow-in to A is ____ 2

The flow across (A,B) is ____ 12 - 2 = 10

?

What is a flow value |f| in this graph? 8 + 2 = 10

Lemma 1

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

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

$\overset{?}{=}0$

Proof.

$$|f| = \sum_{\text{out } s} f(e) \overset{?}{=} \sum_{\text{out } s} f(e) - \sum_{\text{to } s} f(e) =$$

any A

$$= \sum_{v\in A} \left[\underbrace{\sum_{\text{out } v} f(e) - \sum_{\text{to } v} f(e)}_{=0, \text{ except } v=s} \right]$$

by conservation law

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

Lemma 2

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

Proof.

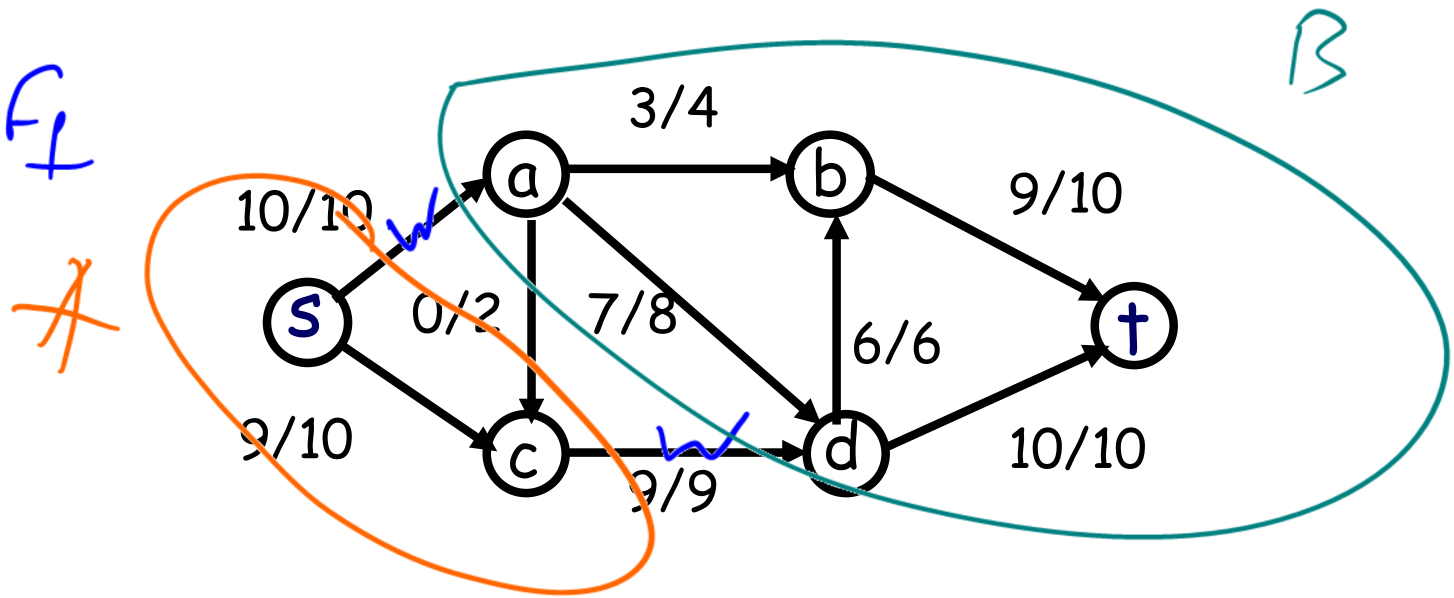
$$|f| \overset{\text{lemma 1}}{=} \sum_{\text{out } A} f(e) - \overbrace{\sum_{\text{to } A} f(e)}^{\geq 0} \overset{?}{\le} \sum_{\text{out } A} f(e)$$

$$\overset{?}{\le} \sum_{\text{out } A} c(e) \underset{\text{def. 2}}{=} cap(A,B)$$

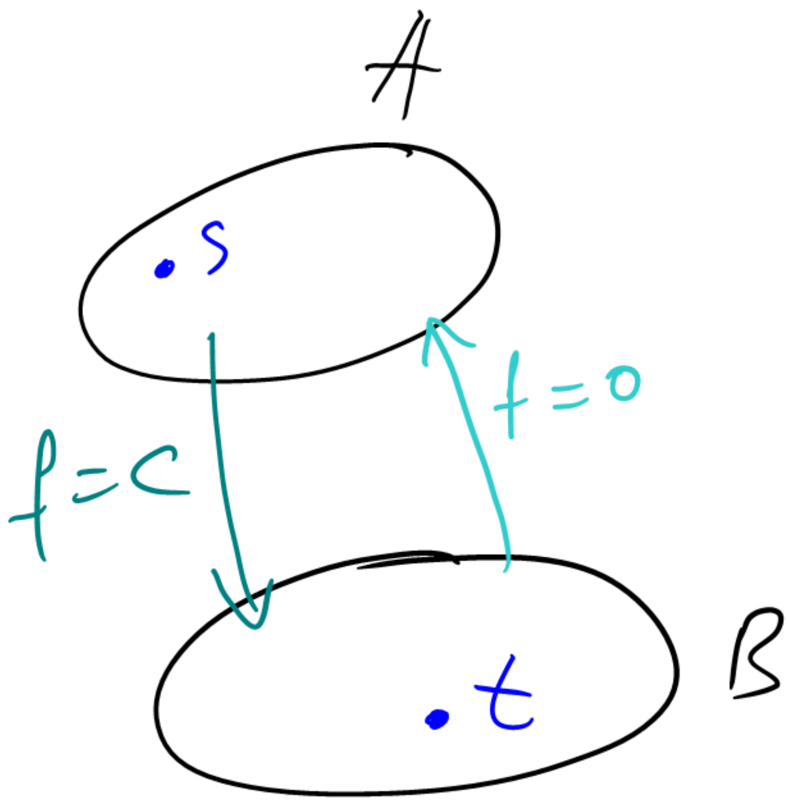
Max-flow Theorem

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

$$\max_f |f| \overset{?}{=} \min_{(A,B)} cap(A,B)$$



Where is a min-cut ?



$\nexists$ s-t path

1. $f=c$ ←
2. $f=0$

page: 116

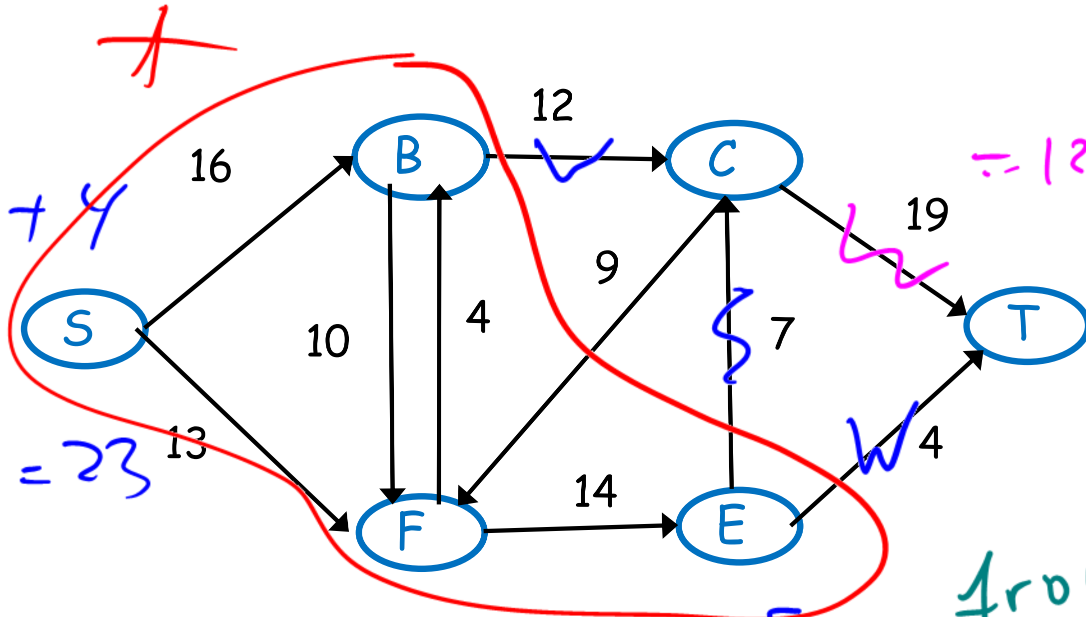
Discussion Problem 1

Run the Ford-Fulkerson algorithm on the following network:

Run FF

1. SBCT, f=12
2. SFET, f=f+4
3. SFECT, 7

max = 12+4+7 = 23



=12+7

① Run FF

② Run traversal on G_f from s

A = {S, B, F, E}

B = {C, T}

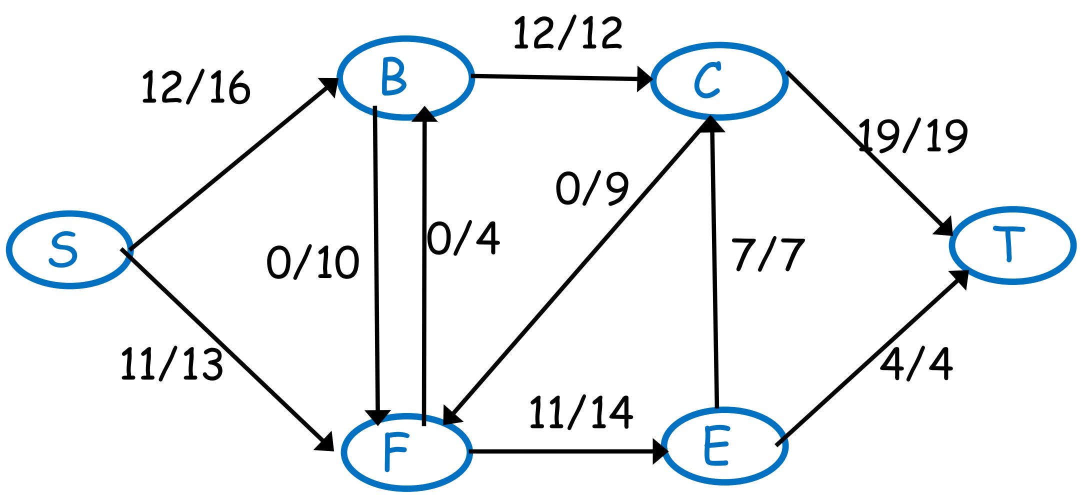
How do you find a min-cut ?

Is a min-cut unique?

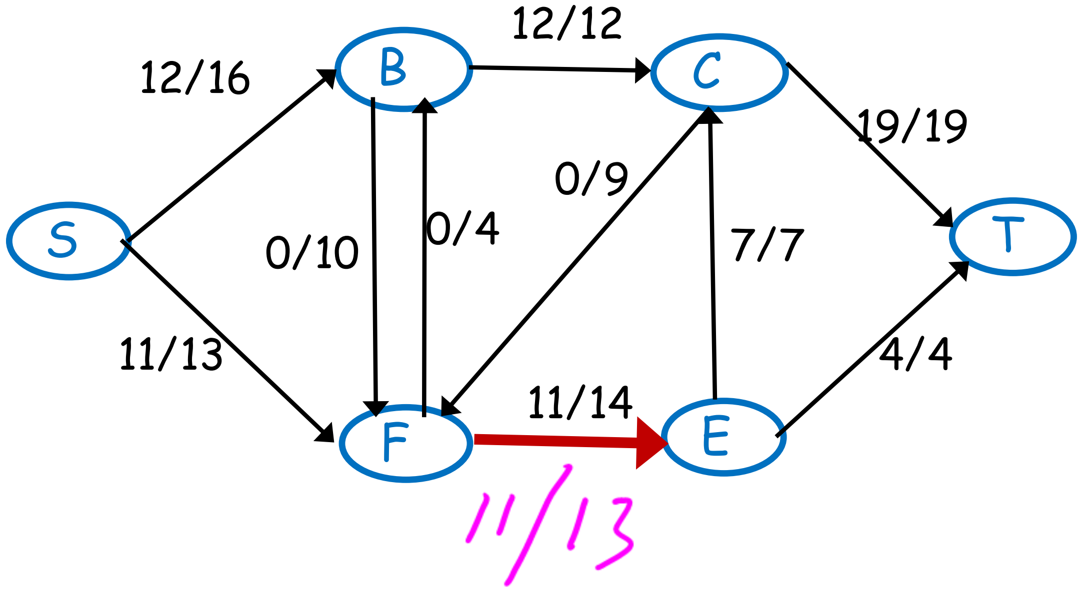
A = {S, B, F, E, C}, B = {T}

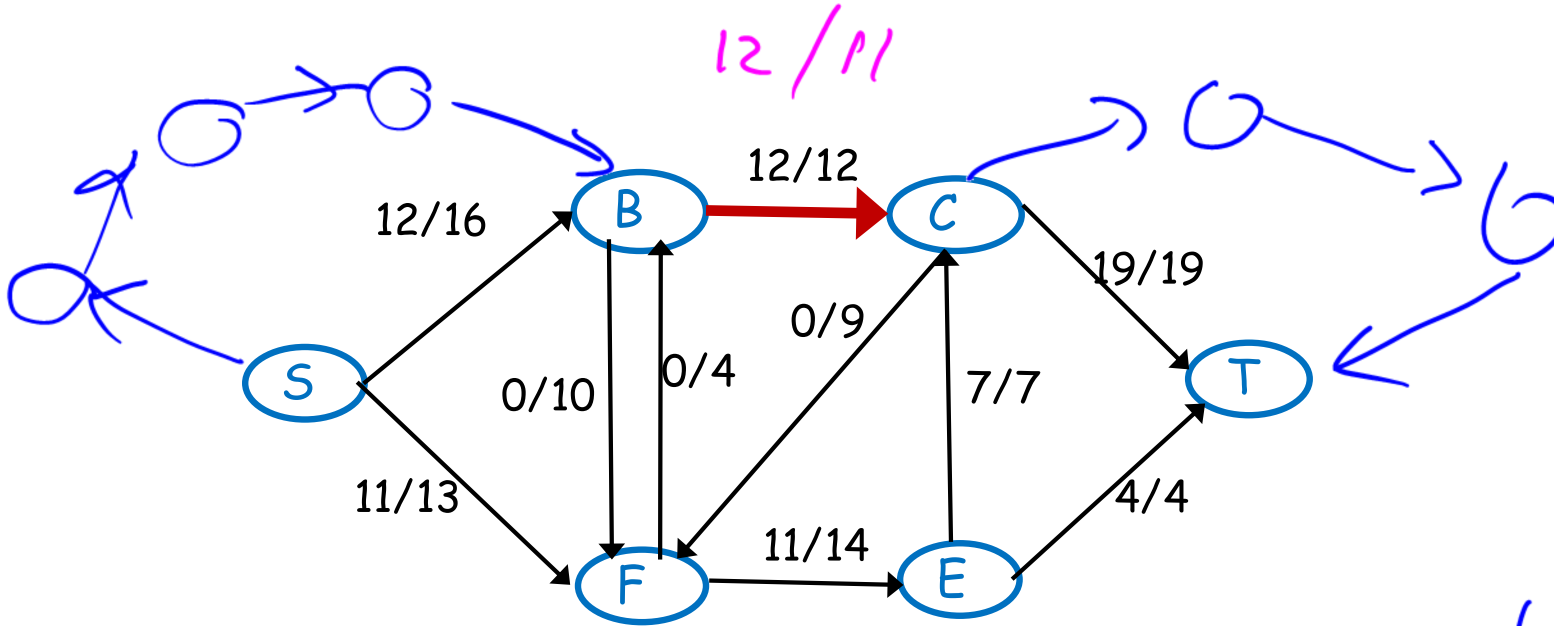
Discussion Problem 2

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.



easy case



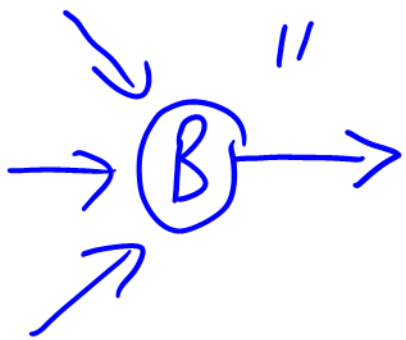


① Find S-B path with not $\emptyset$ flow

② $f^* = f - 1$ along that path

③ Find C-T path

④ $f^* = f - 1$ along that path



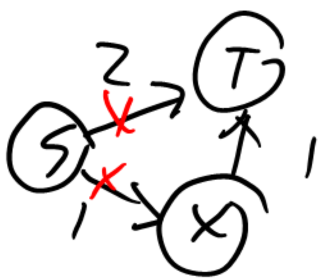
⑤ Run one iteration of FF = find S-T path

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.

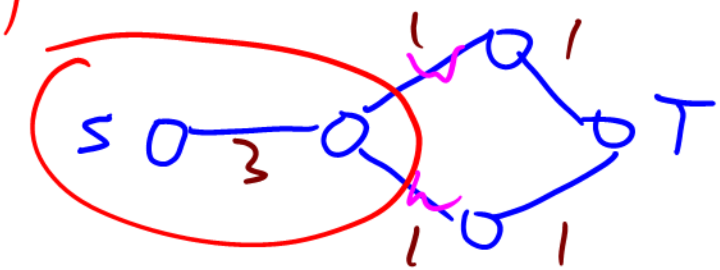
No

max = 3



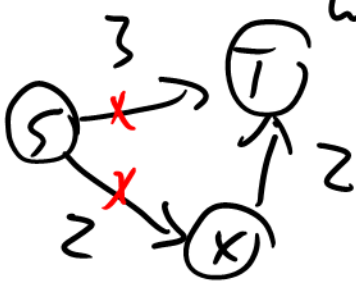
A

max = 2

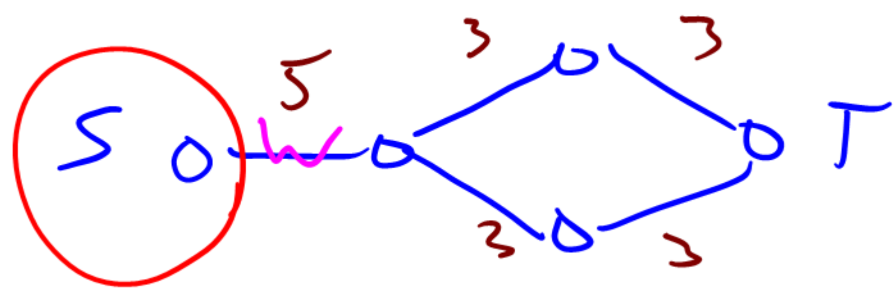


max = 5

max = 5



A

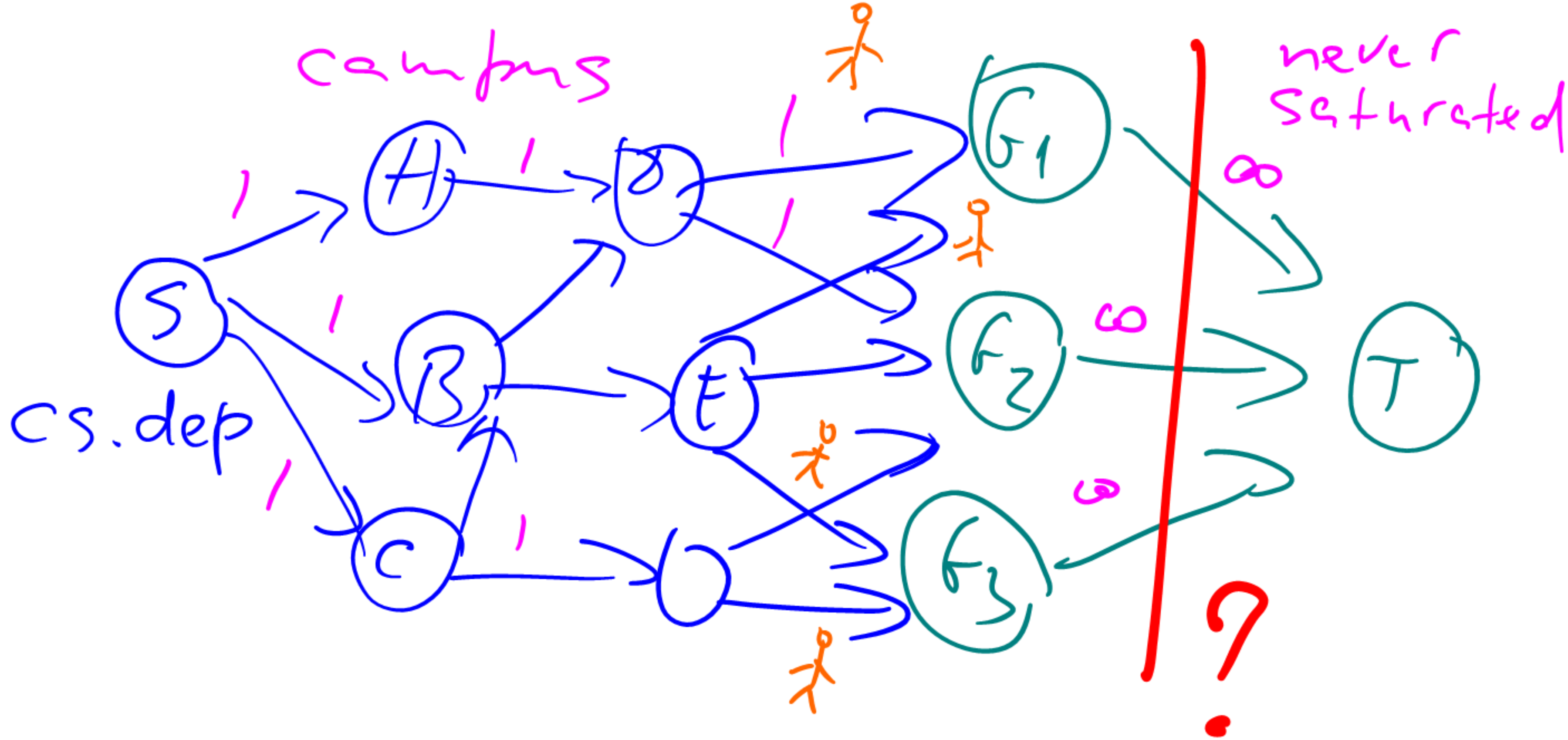


Bad example

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.

= min-cut = max-flow


campus
cs.dep
S
A
B
C
E
G1
G2
G3
T
1
∞
never saturated
?

Reduction

ch. 7.3

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

$\Longleftrightarrow$

- $\forall$ *instance* $y \in Y$ is solvable $\boxed{\text{if and only}}$ if $f(y) \in X$ is solvable.

$$\text{input}(Y) \xrightarrow{f} \text{input}(X)$$

$$\text{solve}(X)$$

$$\text{output}(Y) \xleftarrow{f} \text{output}(X)$$

Solving by reduction to NF

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

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

$\Longrightarrow$

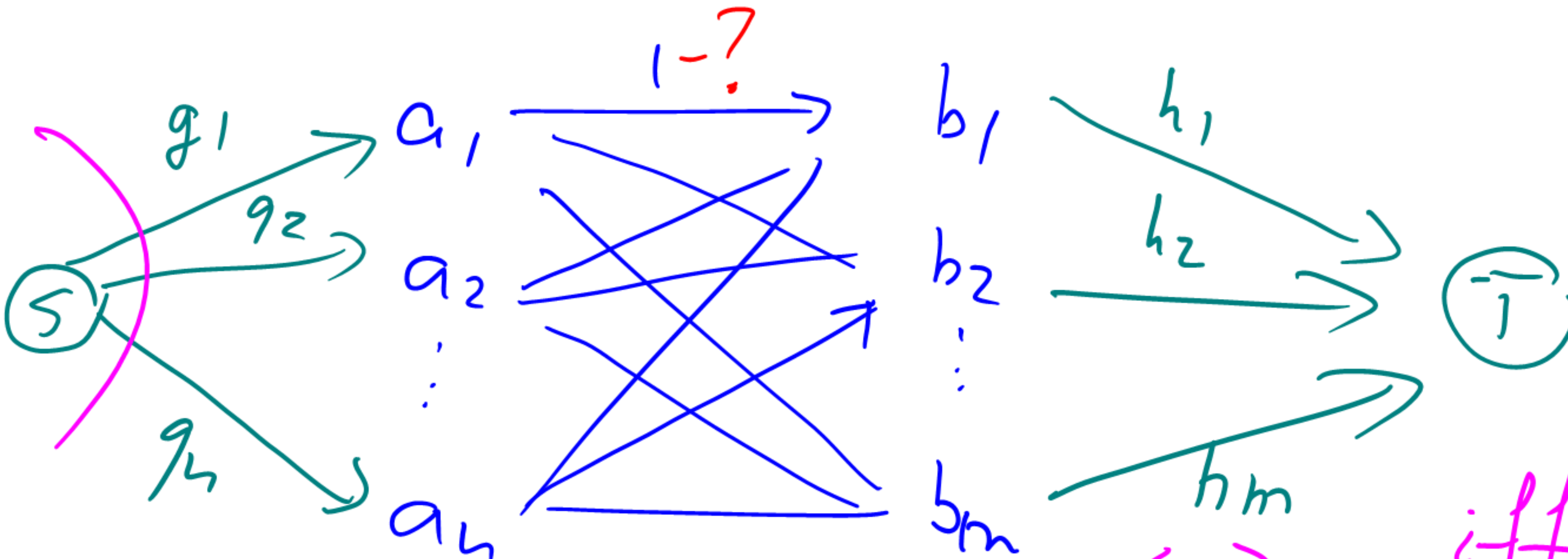
$\Longleftarrow$

Discussion Problem 6

At a dinner party, there are n families a_1, a_2, …, a_n and m tables b_1, b_2, …, 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?

STEP 1

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



$\Longleftrightarrow$, iff

STEP 2

A sitting assignment $\exists$ if and only if the max-flow $= g_1 + g_2 + \dots + g_n$

STEP 3

Proof

$\Rightarrow$ Every member is seated.

we need to prove max-flow $= g_1 + \dots + g_n$

It follows, that edges $s - g_i$ are saturated

So, the flow cannot bigger than

$g_1 + \dots + g_n$.

$\Leftarrow$ Given the max-flow $= g_1 + \dots + g_n$

Find an assignment.

Take saturated edges between g_i and b_j