Analysis of Algorithms

V. Adamchik

Lecture 8

Spring 2023

CSCI 570

University of Southern California

Network Flow

Reading: chapter 7.1 - 7.4

The Network Flow Problem

Our fourth major algorithm design technique

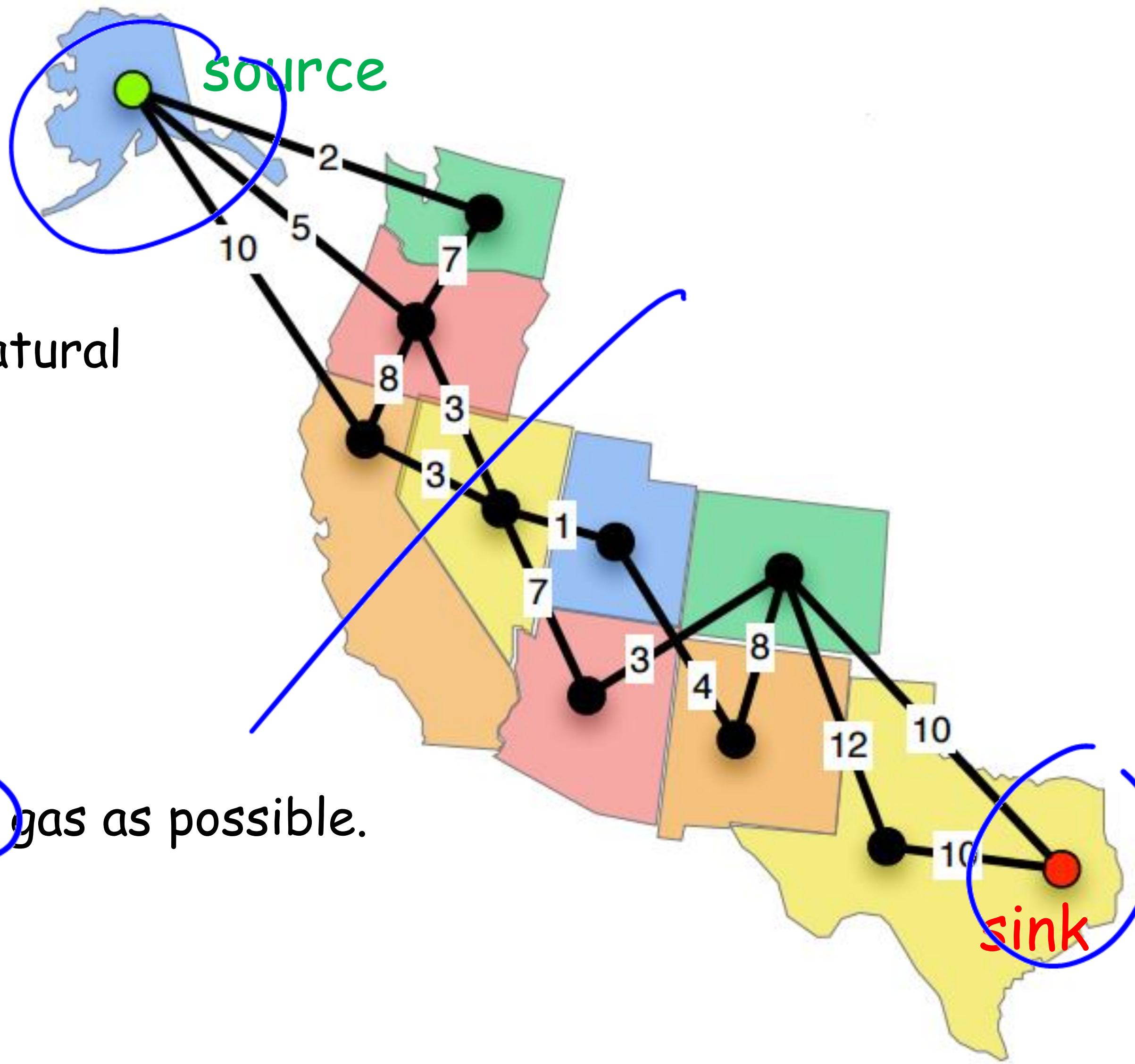
(greedy, divide-and-conquer, and dynamic programming).

Plan:

The Ford-Fulkerson algorithm

Max-Flow Min-Cut Theorem

The Flow Problem



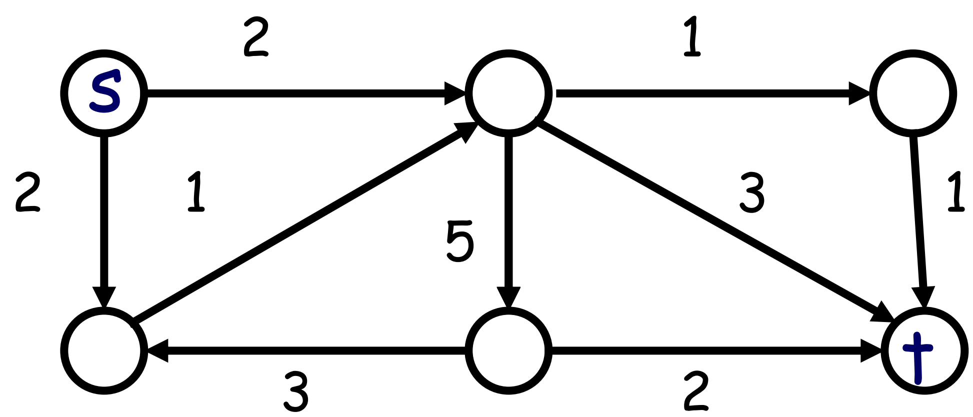
Suppose you want to ship natural gas from Alaska to Texas.

Pipes have capacities.

The goal is to send as much gas as possible.

How can you do it?

The Max-Flow Problem

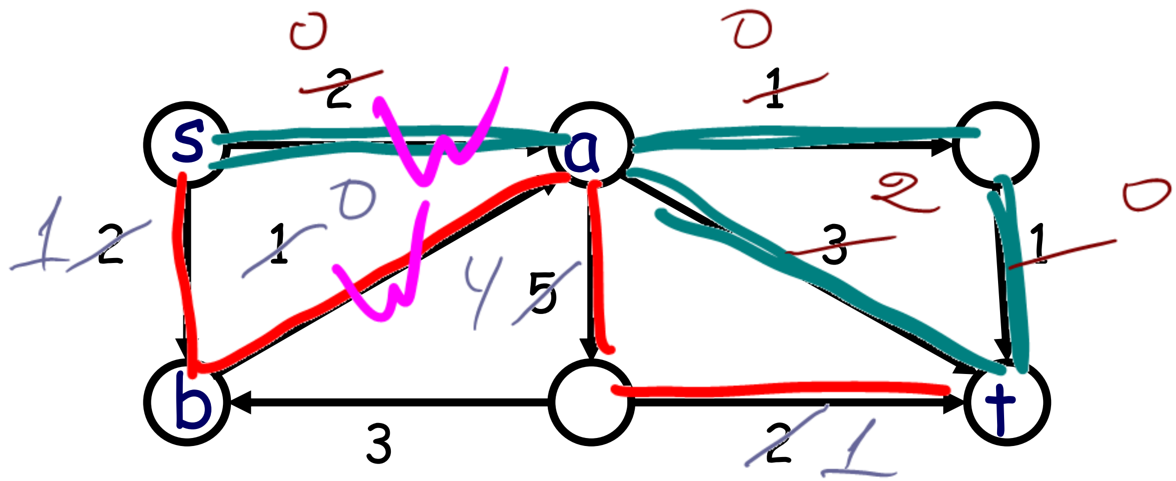


$NF = (V, E, s, t, c)$

we define a *flow* 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: $0 \leq f(e) \leq c(e)$
2. Conservation constraint: $\sum_{\text{into } v} f(e) = \sum_{\text{out } v} f(e)$

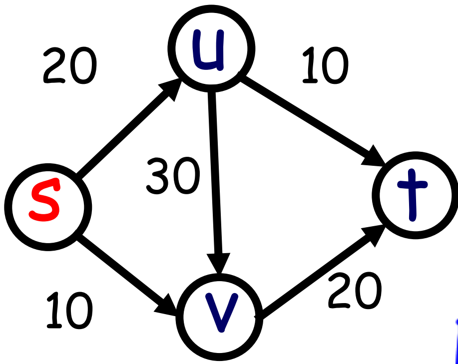
The MAX Flow Problem

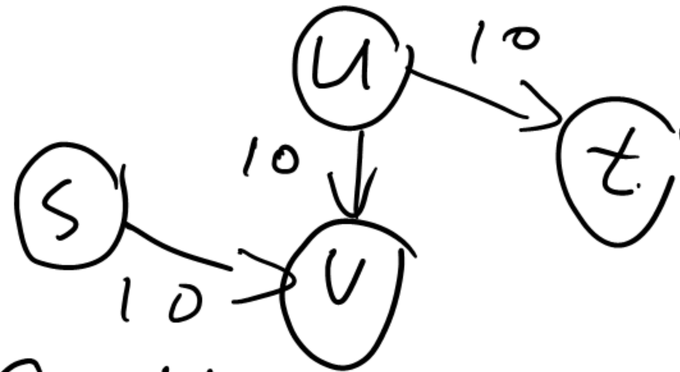


The max-flow here is $\underline{3}$ $= 2 + 1$

How can you see that the flow is really max?

Greedy Approach: push the max





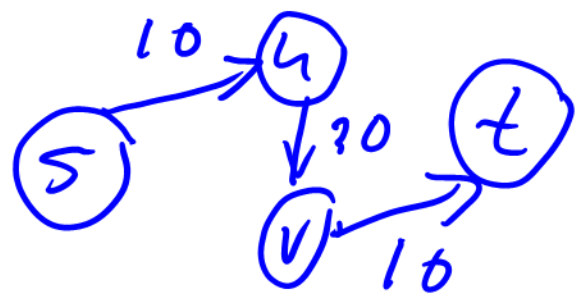
Push 20: s-u-v-t

what is the max?

20

Optimal:

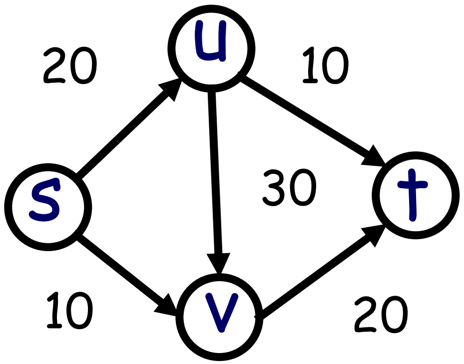
1. Push 10 via s-u-t
2. Push 10: via s-v-t
3. Push 10: s-u-v-t

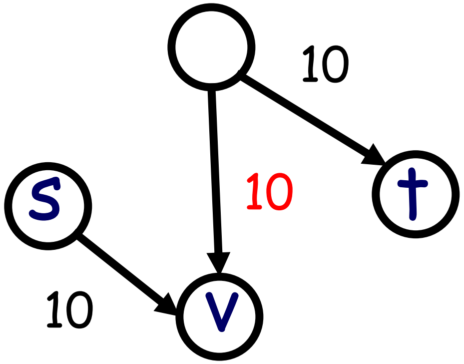


max-flow = 30

Canceling Flow

Push 20 via s-u-v-t





Residual Graph G_f
Residual Capacity c_f

$G = (V, E)$ $\qquad E_f \geq E$

$G_f = (V, E_f)$

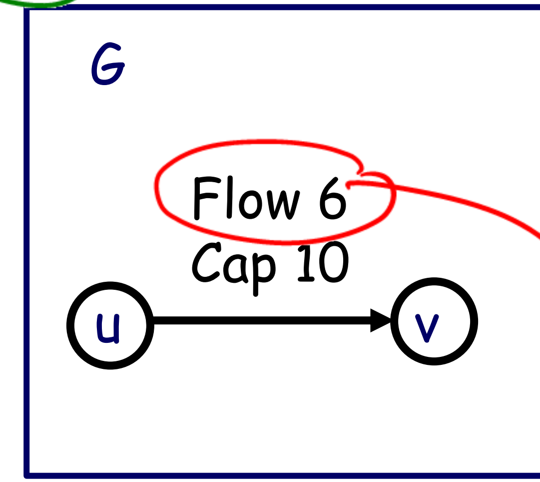
E_f consists of

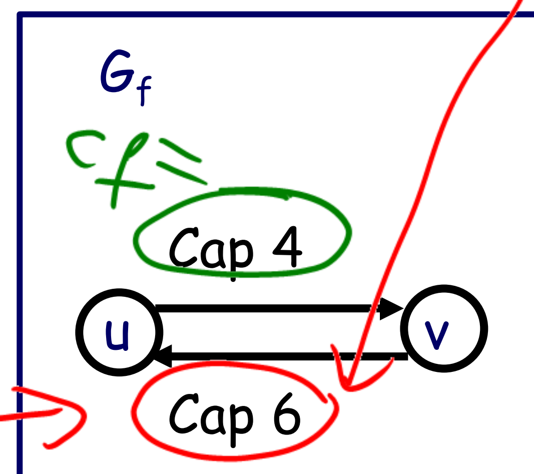
① forward edge (original) $\quad e \in E$

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

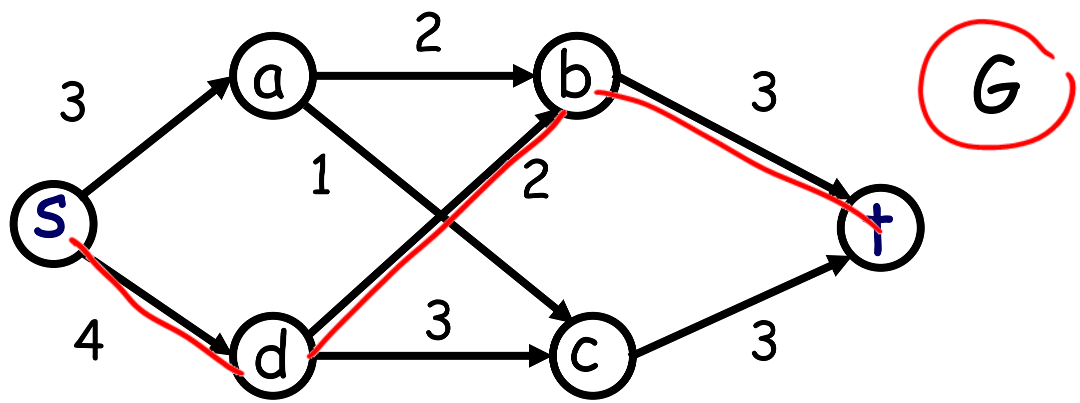
② backward edge $\quad e \notin E$

$c_f = f(e)$

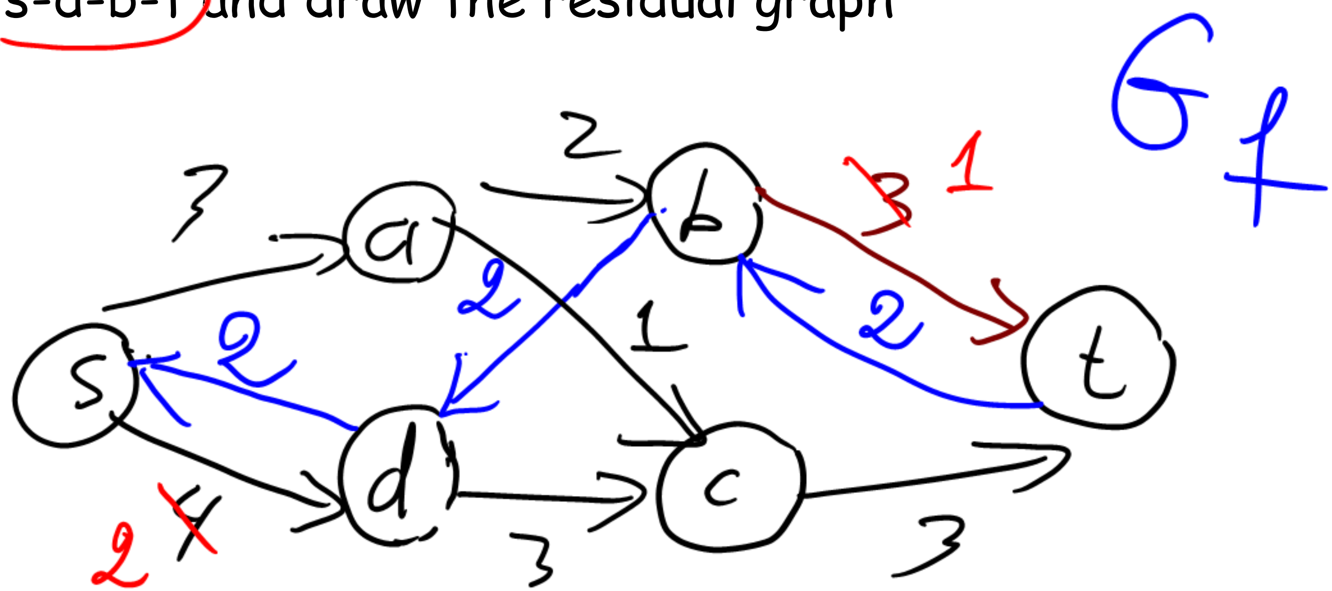




Example: residual graph



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

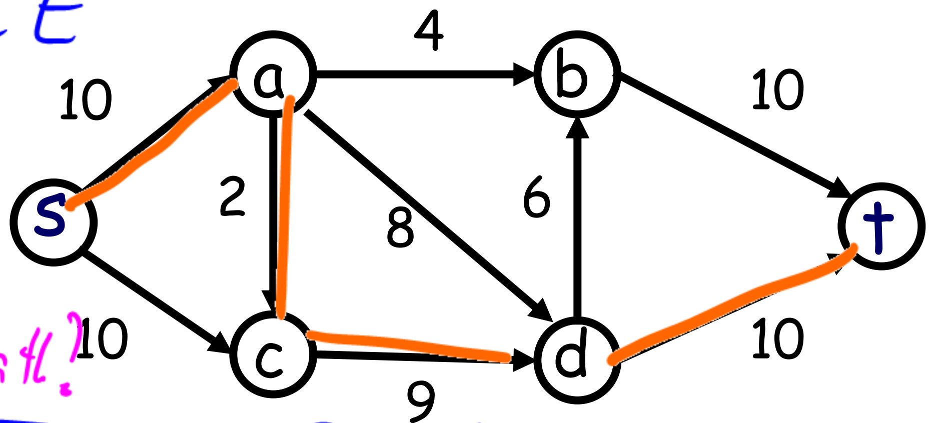
    update the residual graph G_f

$G_f = G$

Example

G

$f(e) = 0, e \in E$



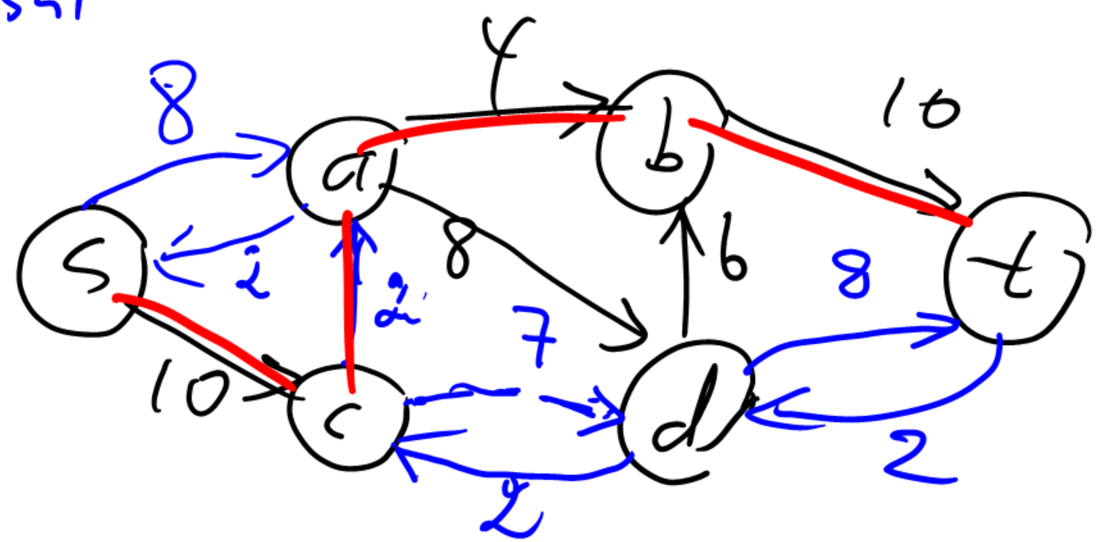
why this path?

Path s-a-c-d-t

any traversal

Push 2

G_f



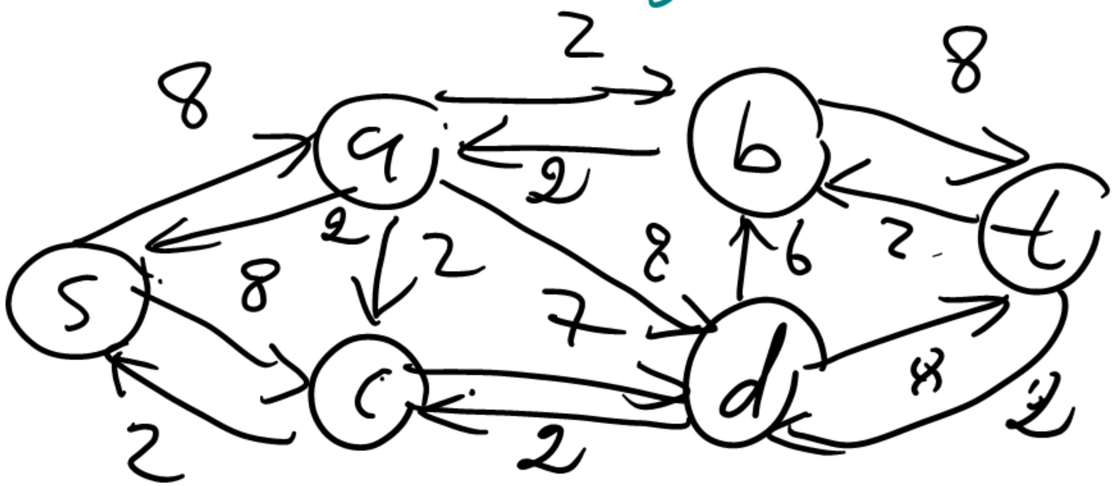
Example

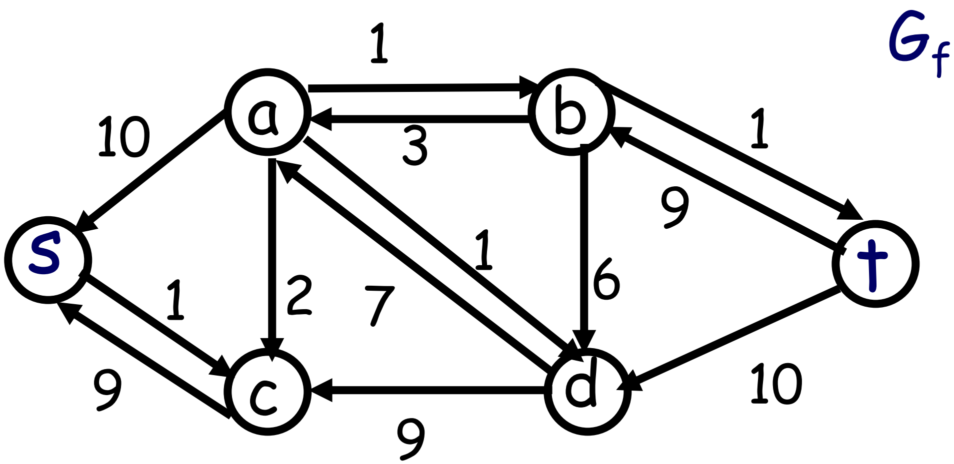
in G_f

Path s-c-a-b-t

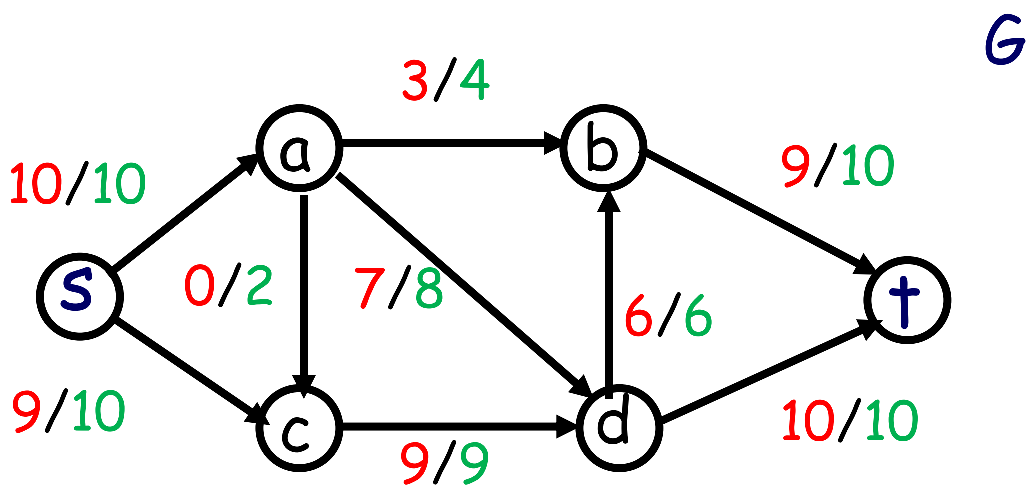
Push 2

Do augmentation!





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 ? traversal $O(V+E)$

find bottleneck $O(V)$

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

update the residual graph $O(V)$

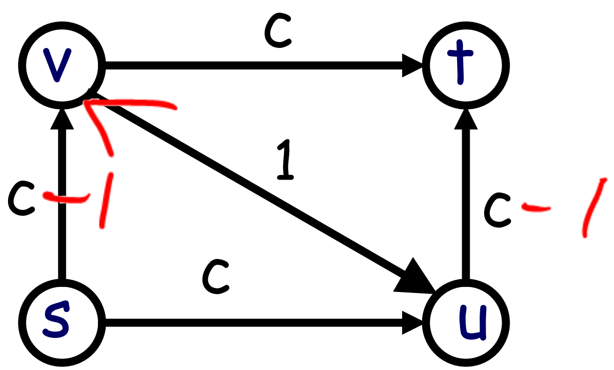
Runtime $= O(V+E) * \#$ of steps

is it polynomial? No

$? \geq \sum_{\text{out } s} c(e)$

The worst-case

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



$c=10^9$

① s-v-u-t, flow = 1

② s-u-v-t, flow = 1+1

repeat

# of iterations = $2 \cdot c$

Proof of Correctness

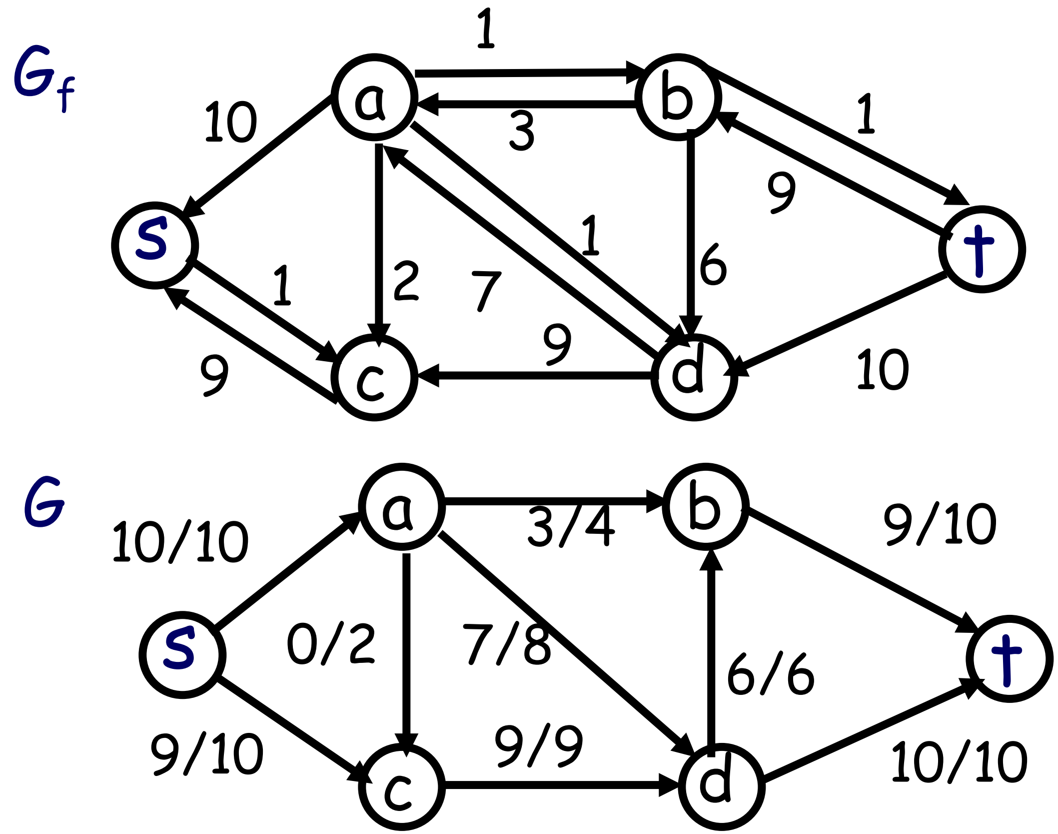
How do we know the algorithm terminate

How do we know the flow is maximum?

2.0001 –

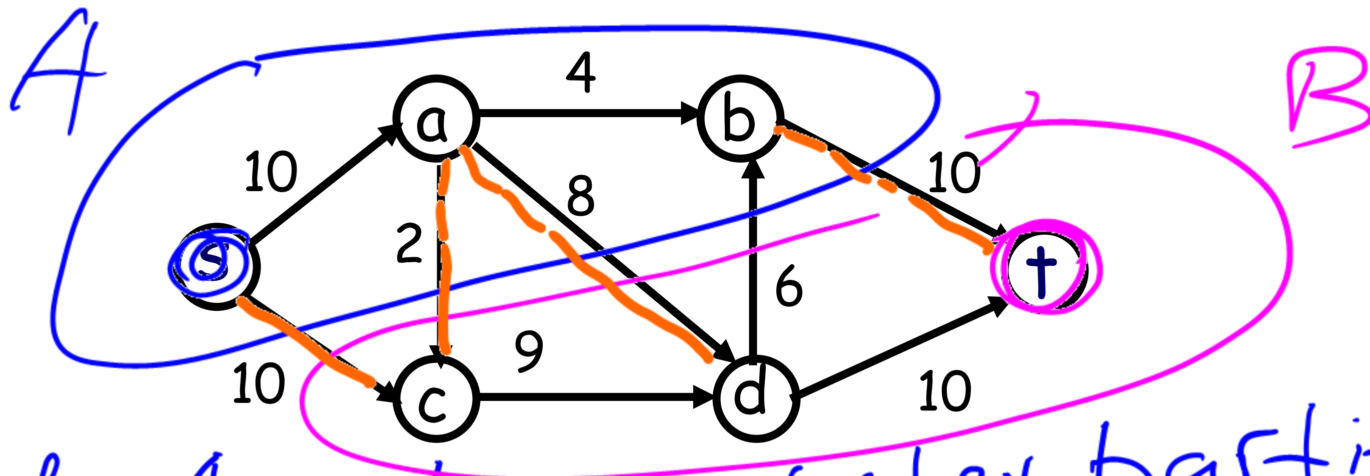
1.99999

= ? = 0



duality

Cuts and Cut Capacity $cap(A,B)$



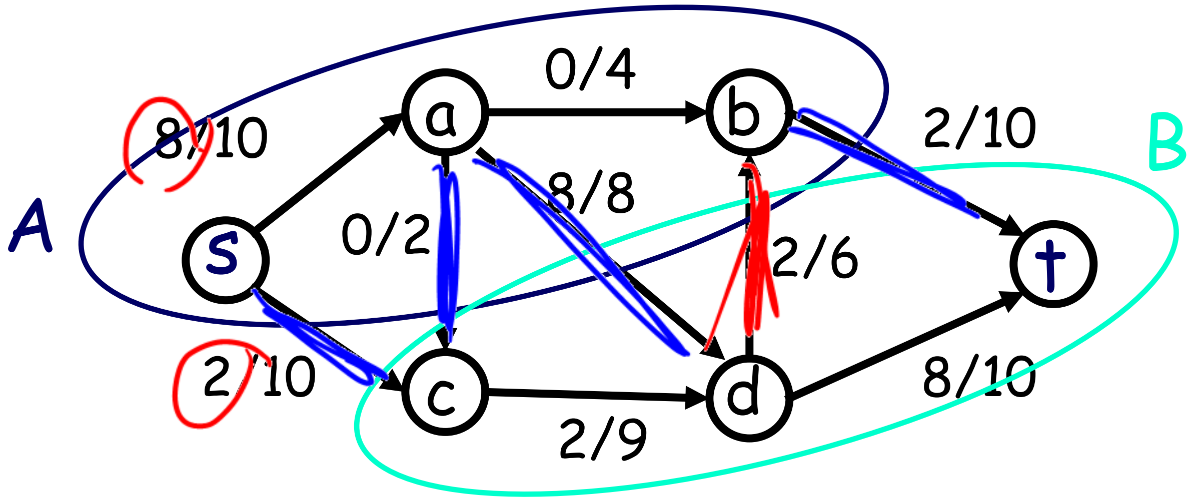
Def1. A cut is a vertex partition, such that $s \in A$, $t \in B$

Def 2. $cap(A,B) = \sum_{\text{out } A} c(e)$

In this example, $cap(A,B) = 10+2+8+10 = 30$

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

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

? $= 0$

Proof.

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

$$\stackrel{?}{=} \sum_{v\in A} \left[\sum_{\text{out } v} f(e) \stackrel{0}{=} - \sum_{\text{to } v} f(e) \right]$$

by conservation law except $v=s$

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

Lemma 2

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

Proof.

by lemma 1 negative ?

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

capacity law ?

$$\le \sum_{\text{out } A} c(e)$$

$$\overset{\text{def}}{=} cap(A,B)$$

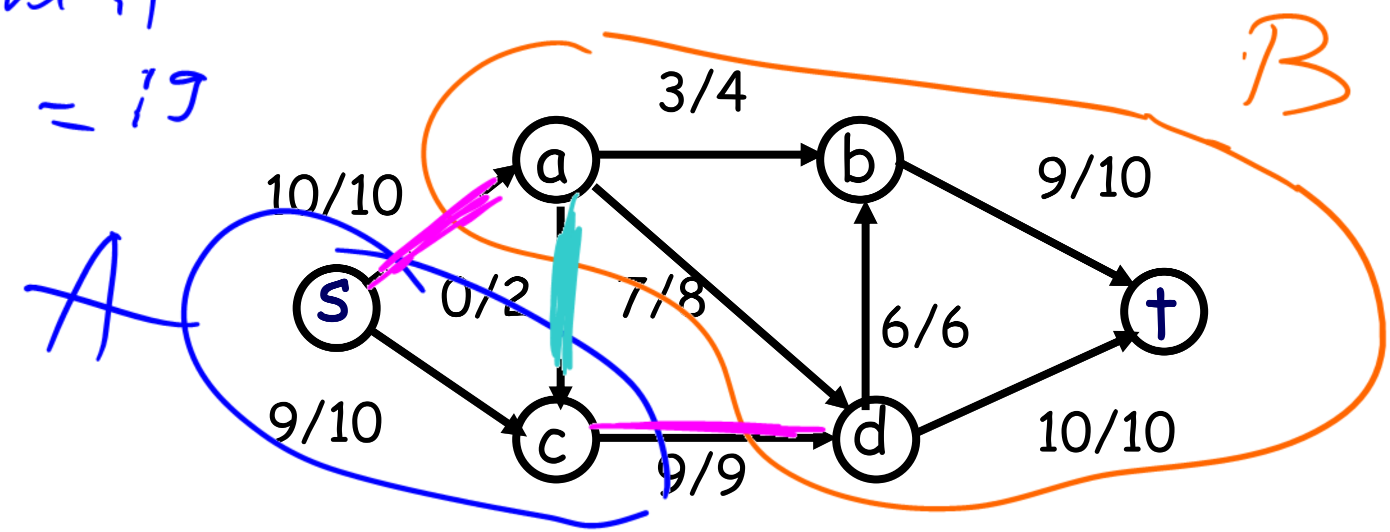
Max-flow Theorem

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

flow/cap

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

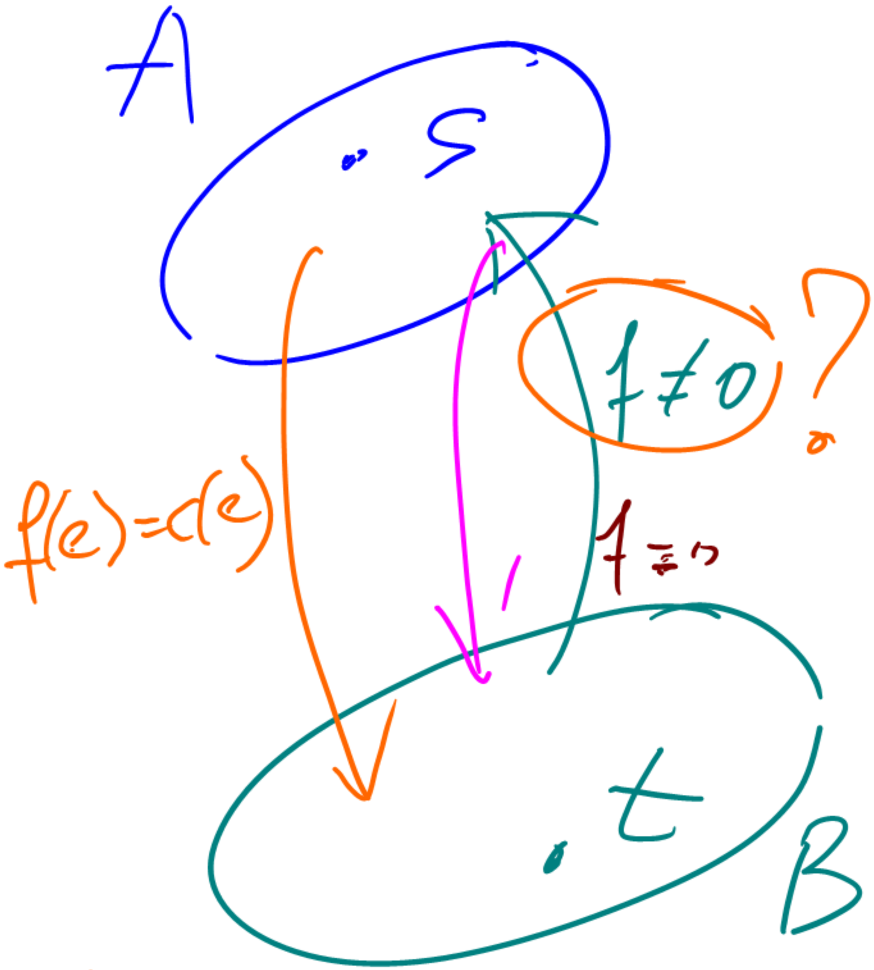
maximal flow = 19



Run traversal from S

Where is a min-cut ?

$\nexists$ s-t path in G_f



proof by condradiction

there must be a flow in opposite direction

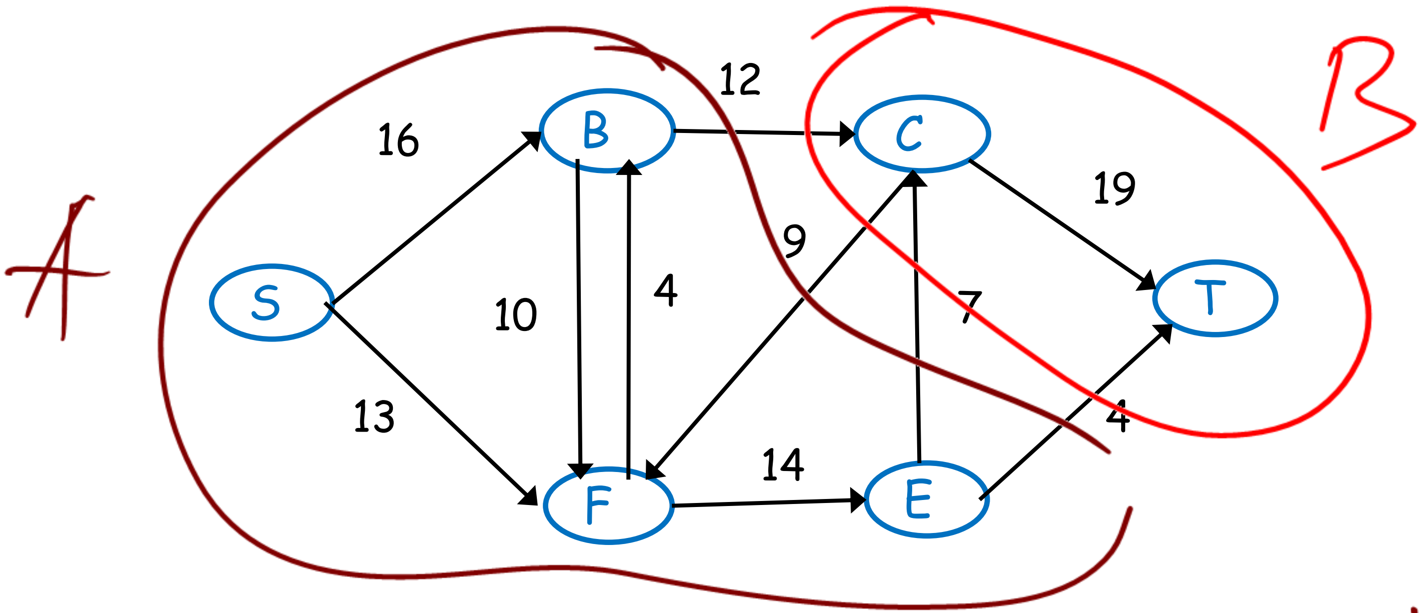
Therefore, we can increase the flow

by condradiction assume $f(e) < c(e)$

read p. 116.

Discussion Problem 1

Run the Ford-Fulkerson algorithm on the following network:



Run a traversal from s in G_f

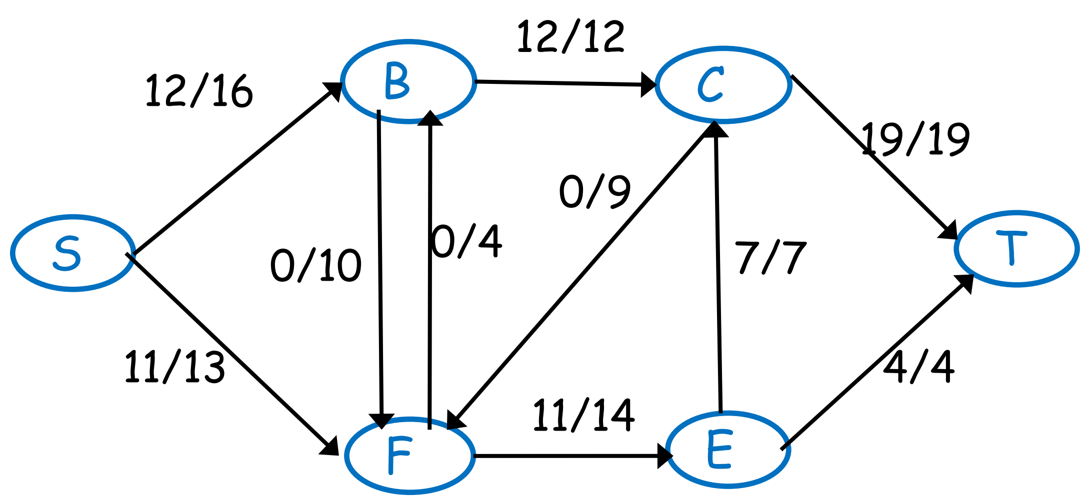
How do you find a min-cut ?

Is a min-cut unique?

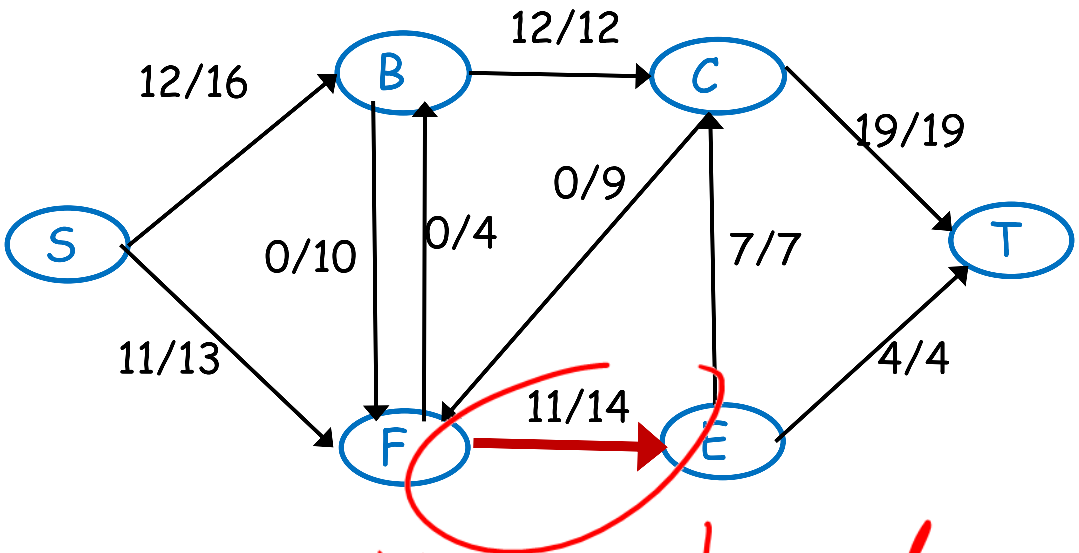
Discussion Problem 2

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.

!

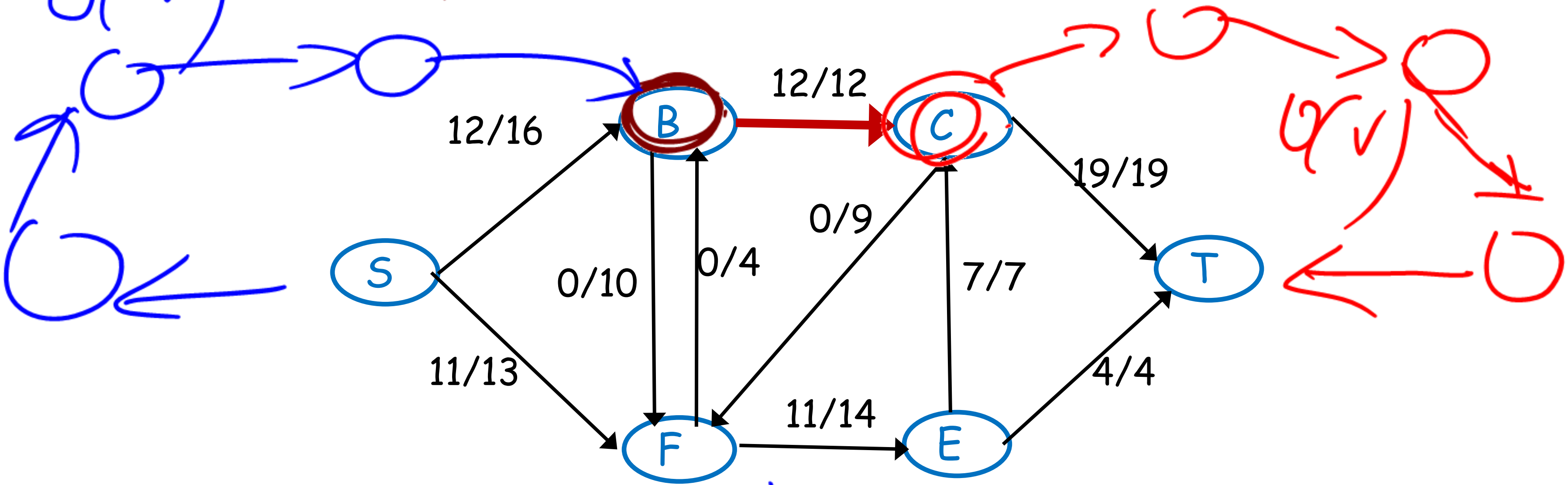


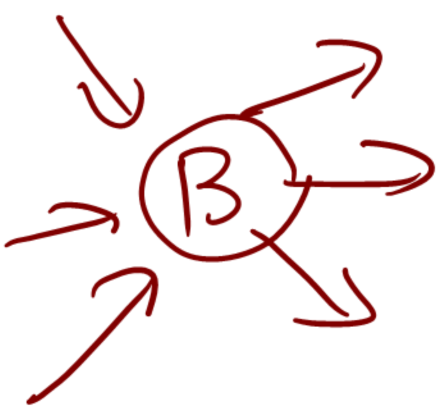
easy case



nothing to do

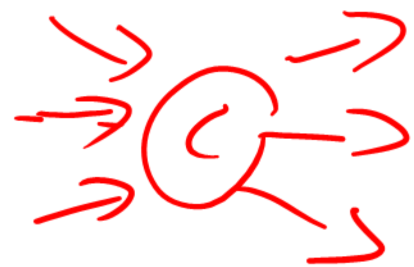
hard case





Algorithm!

1. find S–B path
2. $f^* = f - 1$
3. find C–T path
4. change the flow
5. 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.