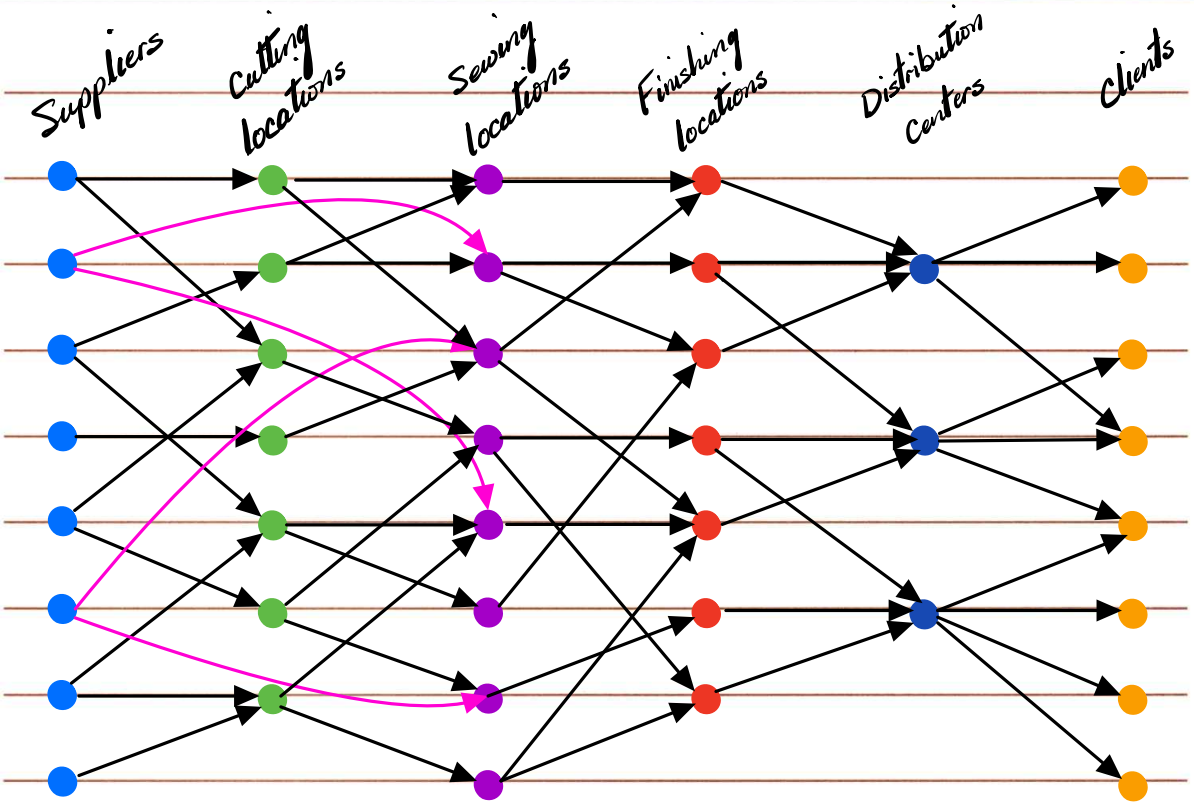
Network Flow

Examples of network flow problems:

- How much data can we send from one node to another node in a computer network?

- How much traffic can the freeways and roads sustain for travel between two cities?

- How profitable can a manufacturing supply chain be?



Sample apparel manufacturing supply chain

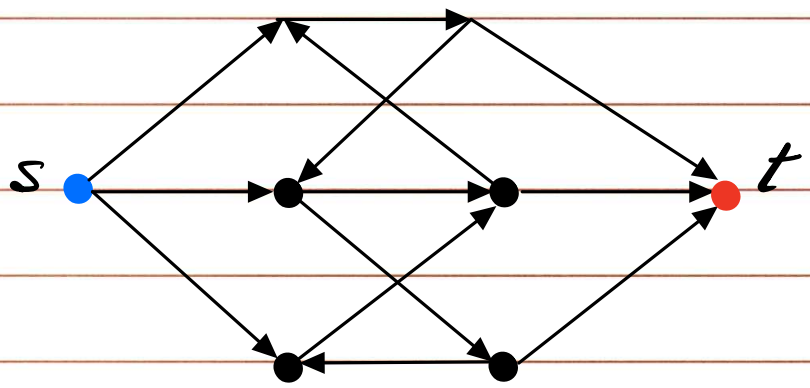
Variables involved in supply chain optimization:

- Supplier capacities
- Supplier costs
- Manufacturing capacities
- Manufacturing costs
- Transportation capacities
- Transportation costs
- DC (warehouse) capacities
- Sales orders and forecasts
- etc.

We will start with simpler networks.

Def. A flow network is a directed graph $G = (V, E)$ with following features:

- Each edge e has a non-negative capacity c_e
- Has a single source node $s \in V$
- Has a single sink node $t \in V$



Sample flow network

Assumptions

- No edges enter s or leave t
- At least one edge is connected to each node $O(m+n) \rightarrow O(m)$
- All capacities are integers

Notation

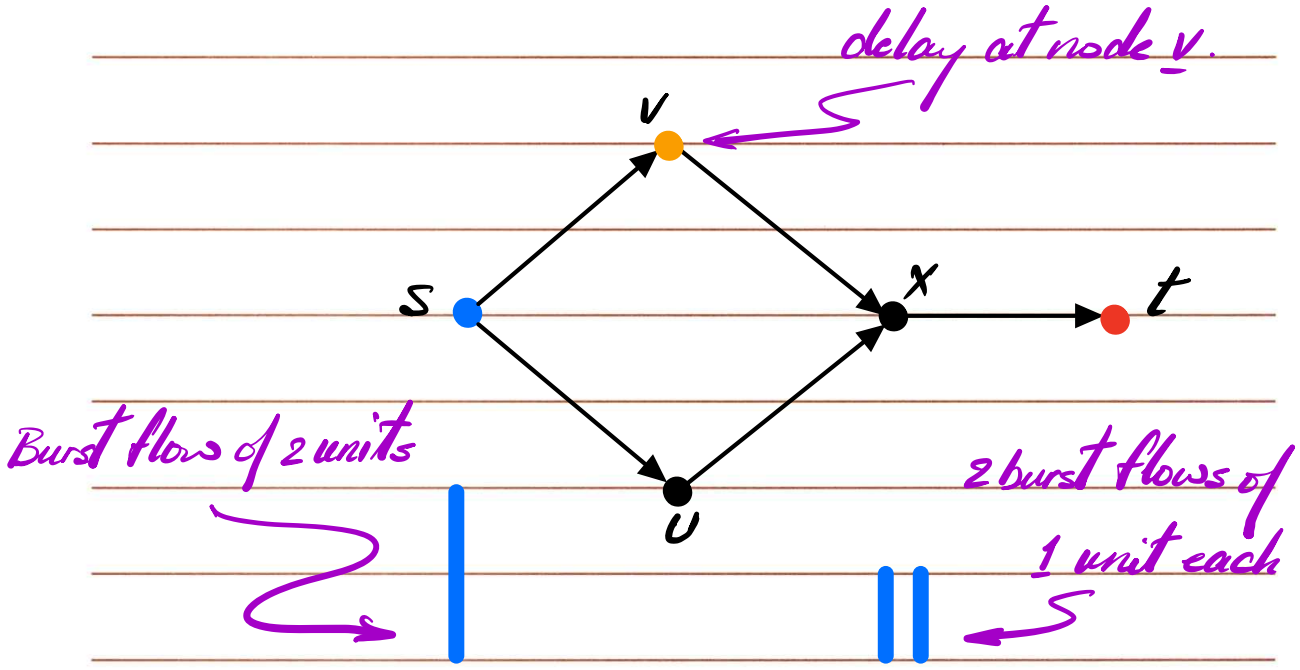
We will call $f(e)$ flow through edge e.
$f(e)$ has the following properties:

1. Capacity Constraint:
   For each edge $e \in E$, $0 \leq f(e) \leq c_e$

2. Conservation of flow:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e) \text{, except for } s \text{ \& } t$$

Our focus is on steady state flow, i.e. no transient or burst flows.

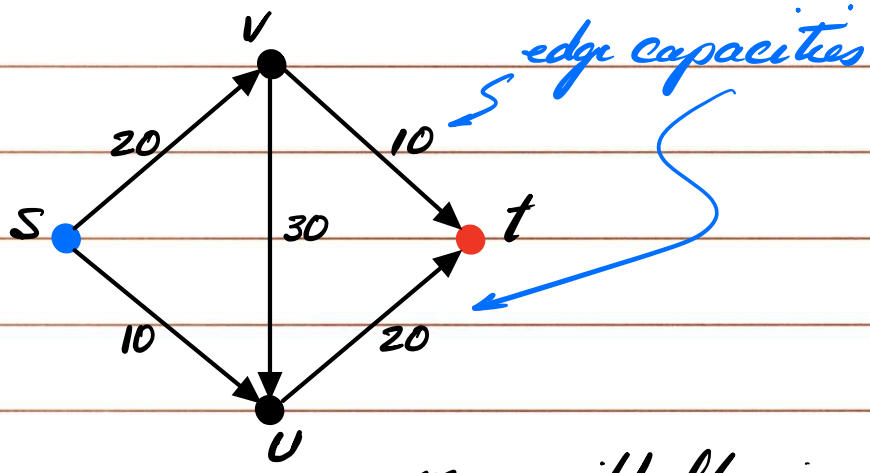


Def For a steady state flow, the value of flow $v(f)$ is defined as follows:

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

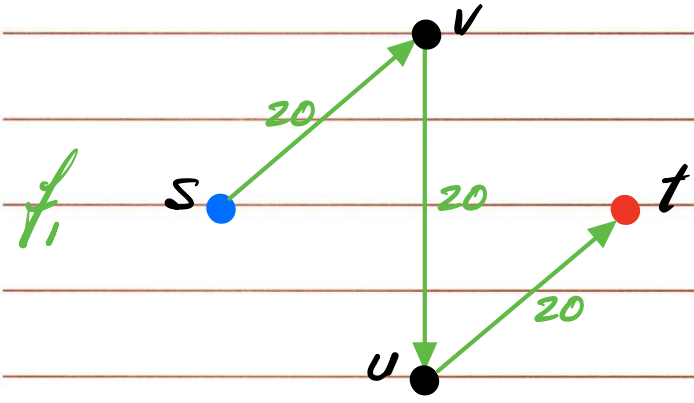
Max Flow Problem Statement

Given a flow network G, find an s-t flow with maximum value.



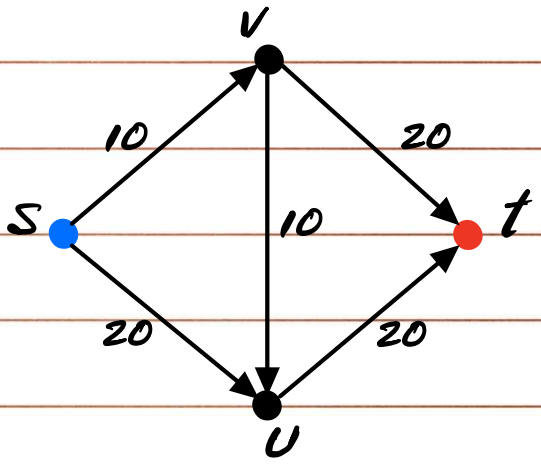
max. possible flow is 30 units

Try #1 Use highest capacity edges first



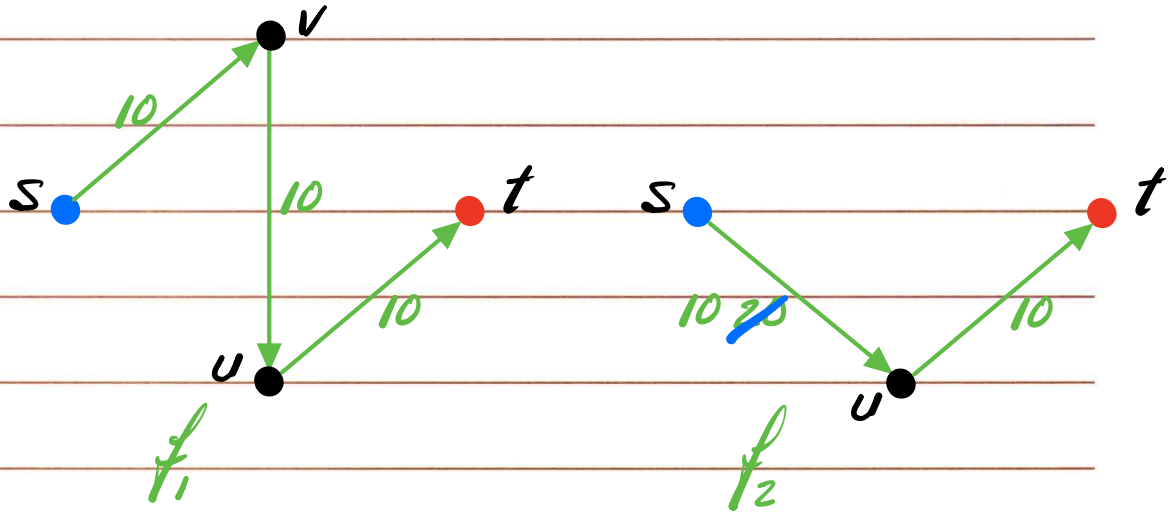
Any more flow pushed out on su will get stuck at u. We are therefore unable to find max possible flow of 30 units

Try #2 Use lowest capacity edges first.



Counter example

Max. possible flow is 30 units



Any flow more than 10 units pushed out on su will get stuck at u. We are therefore unable to find max possible flow of 30 units

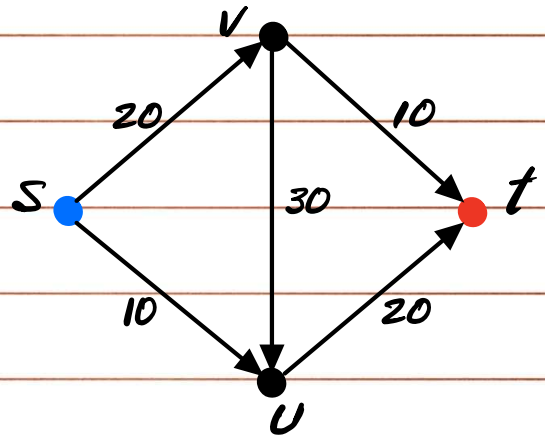
Def. G_f is the residual graph of G with flow f the following definition:

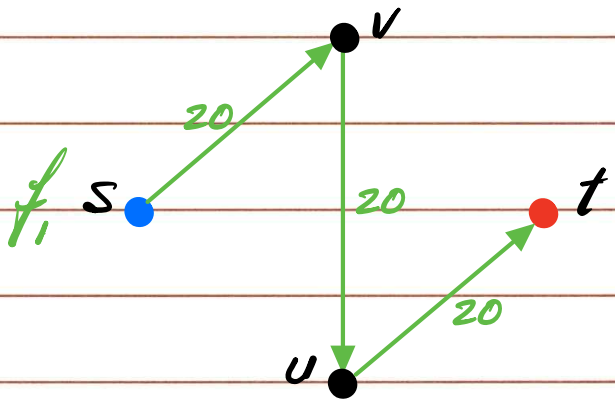
- G_f has the same number of nodes as G
- forward edges: For each edge e with $f(e) < c_e$, we include edge e in G_f with capacity $c_e - f(e)$
- backward edges: For each edge e with $f(e) > 0$, we include edge e' (opposite direction to e) in G_f with capacity $f(e)$

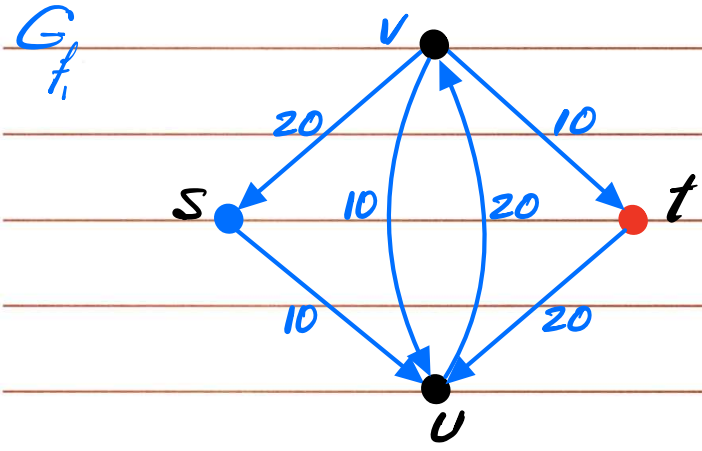
To create G_f:

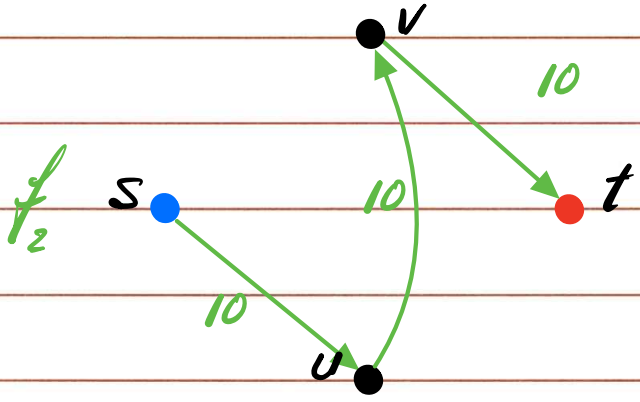
- If $f(e) = 0$
  G_f will have 1 forward edge w/ cap. c_e
- If $f(e) = c_e$
  G_f will have 1 backward edge w/ cap. c_e
- If $0 < f(e) < c_e$
  G_f will have 1 forward edge w/ cap. $c_e - f(e)$, and 1 backward edge w/ cap. $f(e)$

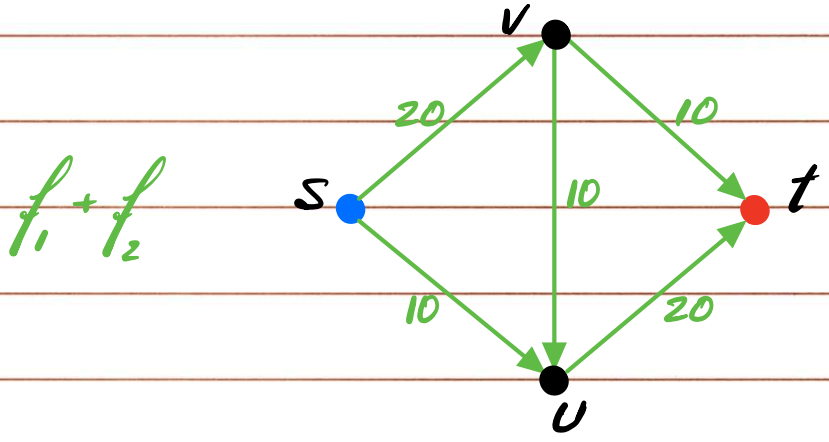
Same as Try #1, but w/ residual graphs.











value of flow = 30

We have reached max. flow!

Def. If P is a simple path from s to t in G_f, then bottleneck(P) is the min. residual capacity of any edge on P.

High level strategy to find max flow:

- Find a path from s to t
- Find the bottleneck value for this path
- Push flow through this path with value equal to bottleneck value.
- Repeat

```
Augment(f, C, P)
  Let b = bottleneck(P)
  for each edge (v,u) ∈ P
    if e = (v,u) is a forward edge
      increase f(e) by b units
    else (v,u) is a backward edge
      let e = (u,v)
      decrease f(e) by b units
    endif
  endif
```

Return (f')

If f is flow before augmentation and f' is flow after augmentation, we need to show that if f is a valid flow, then f' will also be a valid flow.

Proof: 1- Check capacity condition

Need to show that for each edge $e \in E$, we have $0 \leq f'(e) \leq c_e$

A- If e is a forward edge, then

$f(e) + \text{bottleneck}(P) \leq c_e - f(e) + f(e)$

$0 \leq f'(e) \leq c_e$

B- If e is a backward edge, then

$\text{bottleneck}(P) \leq f(e)$

$f(e) - \text{bottleneck}(P) \geq f(e) - f(e)$

$c_e \geq f'(e) \geq 0$

2- Check conservation of flows

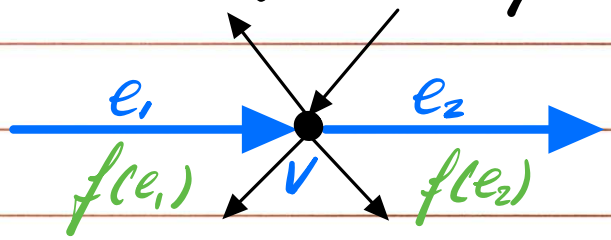
Since f is a valid flow, for each node v (other than s and t), we have

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

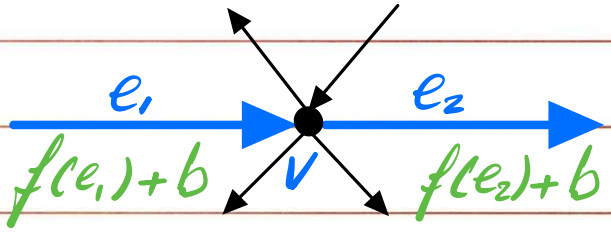
Consider a node v on augmentation path P.

Case 1- Both edges e_1 and e_2 incident on v and belonging to P, correspond to forward edges in G_f

3 other cases left as an exercise



before augmentation



after augmentation

$$\sum_{e \text{ into } v} f(e) + b = \sum_{e \text{ out of } v} f(e) + b$$

Ford-Fulkerson Algorithm

MaxFlow (G, s, t, C)
Initially $f(e) = 0$ for all e in G
While there is an s-t path in G_f
    Let P be a simple s-t path in G_f
    $f' =$ Augment (f, C, P)
    $f = f'$
    update G_f
endWhile
Return f

Proof of correctness should include

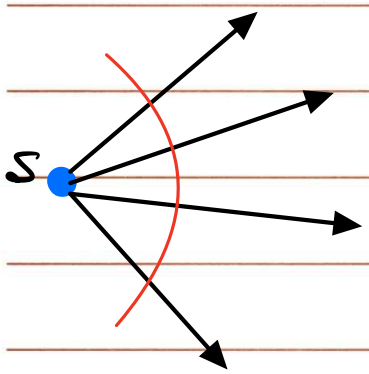
- Proof of termination
- Proof of optimality
  (i.e. f is a Max Flow)

1- While loop terminates

a- f is an integer flow

Proof by math. induction

b- $v(f)$ increases by at least 1 unit at each iteration

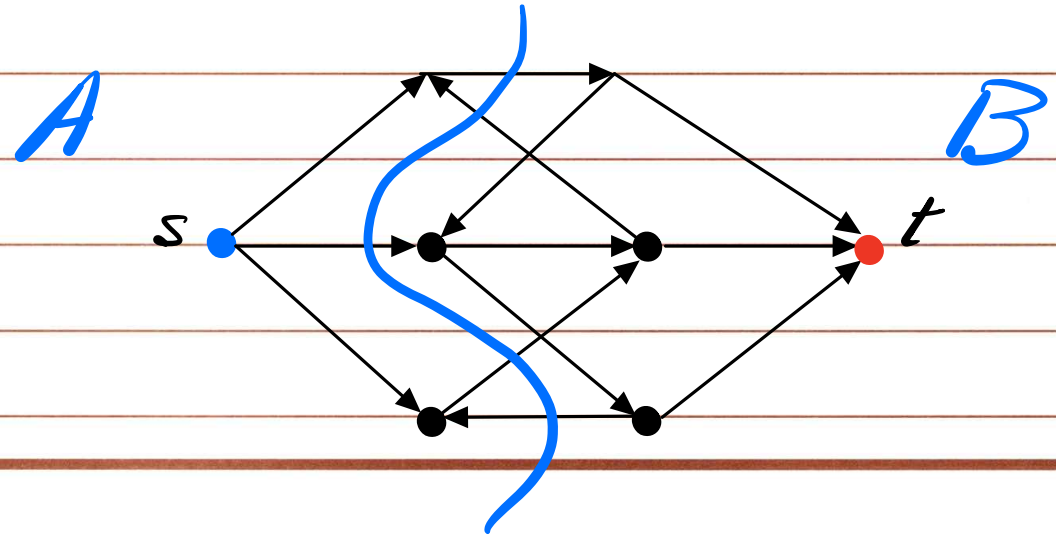


$$v(f) = \sum_{e \text{ out of } s} f(e) \leq \sum_{e \text{ out of } s} c_e = C$$

So, the while loop must terminate in at most C iterations

2- f is a Max Flow

Def. A cut divides nodes in the graph into 2 sets A and B such that $s \in A$ and $t \in B$

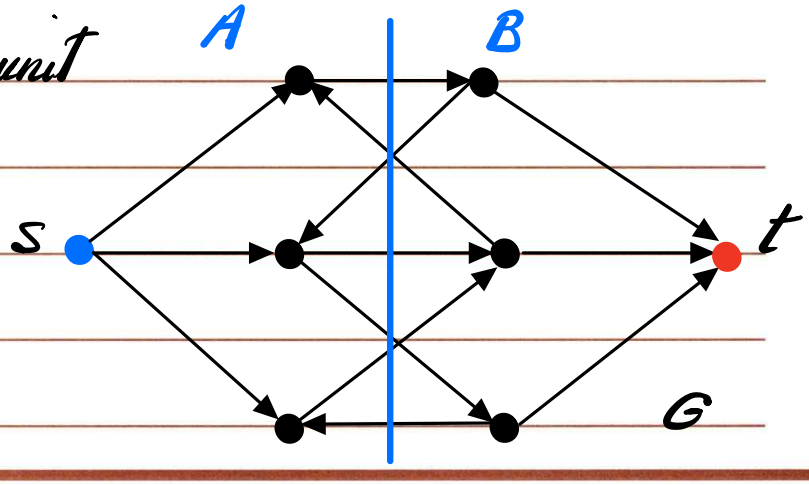


Def. The capacity of the cut (A,B), denoted $C(A,B)$ is the total capacity of edges out of A.

$$C(A,B) = \sum_{e \text{ out of } A} c_e$$

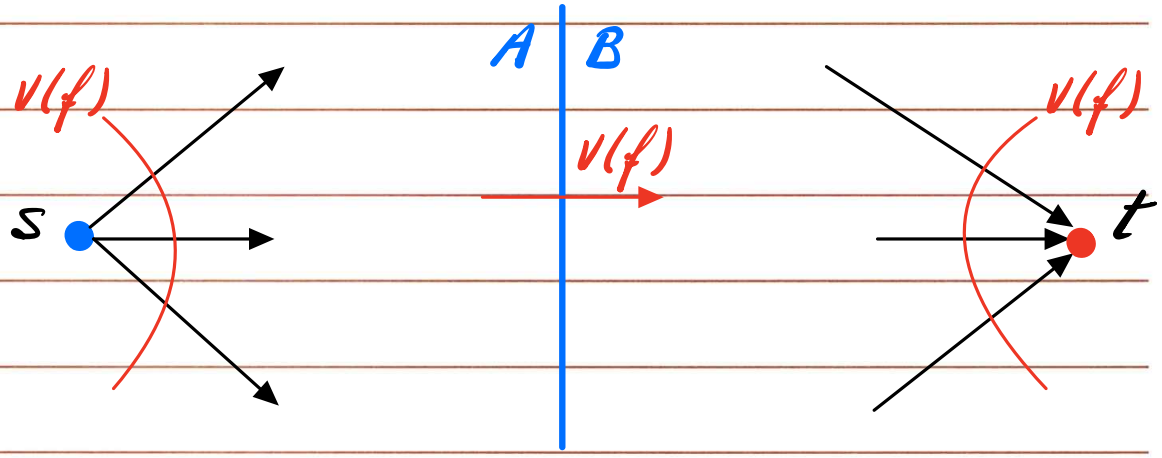
If all edges have unit capacity, what is the capacity of this (A,B) cut?

4



Fact: Let f be any s-t flow and (A,B) any s-t cut, then

$$v(f) = f^{out}(A) - f^{in}(A)$$



Fact:

Max. value of the flow $\leq$ Cap. of the (A,B) cut

Proof:

$$v(f) = f^{out}(A) - f^{in}(A)$$

$$v(f) \leq f^{out}(A) \leq \sum_{e \text{ out of } A} c_e$$

$$v(f) \leq C(A,B)$$

Cut (A, B) could be any cut, so Max flow is bounded by the capacity of every s-t cut.

Recall that Ford-Fulkerson terminates when the flow f has no s-t path in G_f

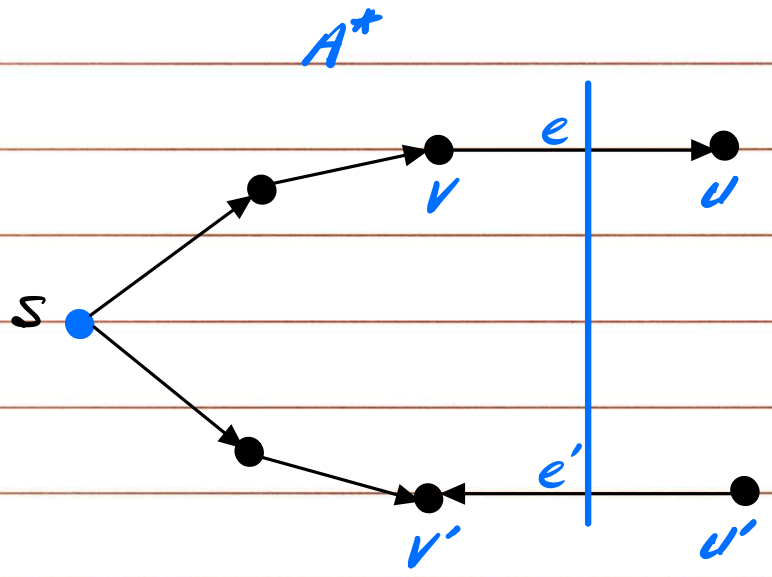
Claim: If there is no s-t path in G_f, then there is an s-t cut (A^*, B^*) where

$$v(f) = c(A^*, B^*)$$

Proof:

Create sets A^* and B^* such that A^* includes all nodes v where there is an s-v path in G_f.

Then $B^* = V - A^*$



$f(e) = c_e$

$f(e') = 0$

G

$$v(f) = f^{out}(A^*) - f^{in}(A^*)$$

$$v(f) = \sum_{e \text{ out of } A^*} c_e - 0$$

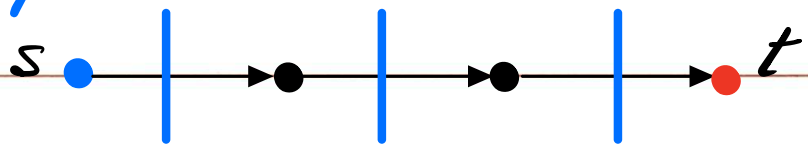
$$v(f) = C(A^*, B^*)$$

$v(f)$ = Max value of any s-t flow in G.

And, (A^*, B^*) has the min. capacity of any s-t cut in G.

Q: Is the min. cut unique? No!

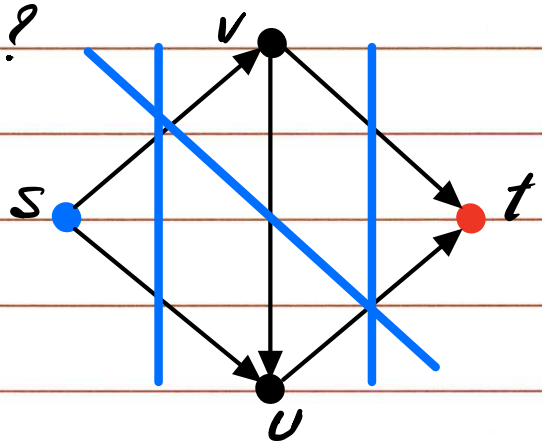
ex.: all edges have unit capacity in below graph



Here we have 3 min cuts.

Q: How many min cuts are there in the following graph with unit capacity edges?

3



Q: How can we determine if the min cut is unique?

Find min cut closest to s
Find min cut closest to t
If they are the same cut, then the min cut must be unique.

Complexity Analysis:

MaxFlow (G, s, t, C)

Initially $f(e) = 0$ for all e in G

While there is an s-t path in G_f

O(m) Let P be a simple s-t path in G_f

O(m) $f' =$ Augment (f, c, P)

O(m) $f = f'$

O(m) update G_f

endWhile

Return f

O(C)

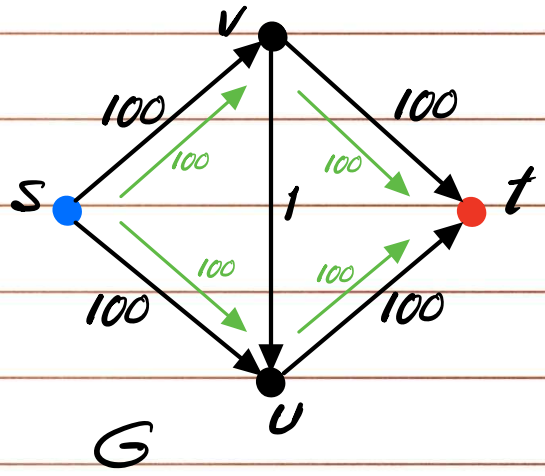
Overall complexity = $O(Cm)$

Q: Is this an efficient solution?

No! (because of C)

How many iterations of the Ford-Fulkerson algorithm does it take to find max. flow in G?

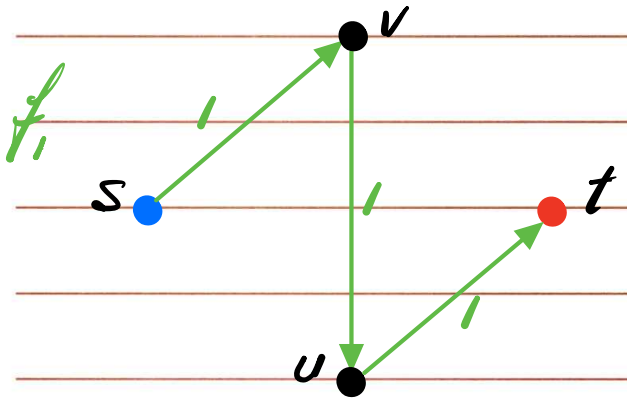
A- In the best case: 2 iterations

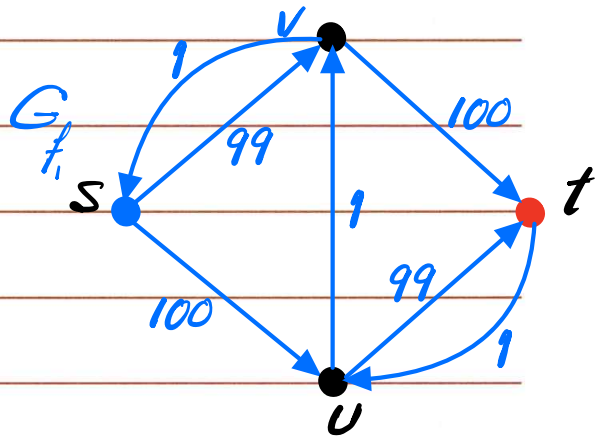


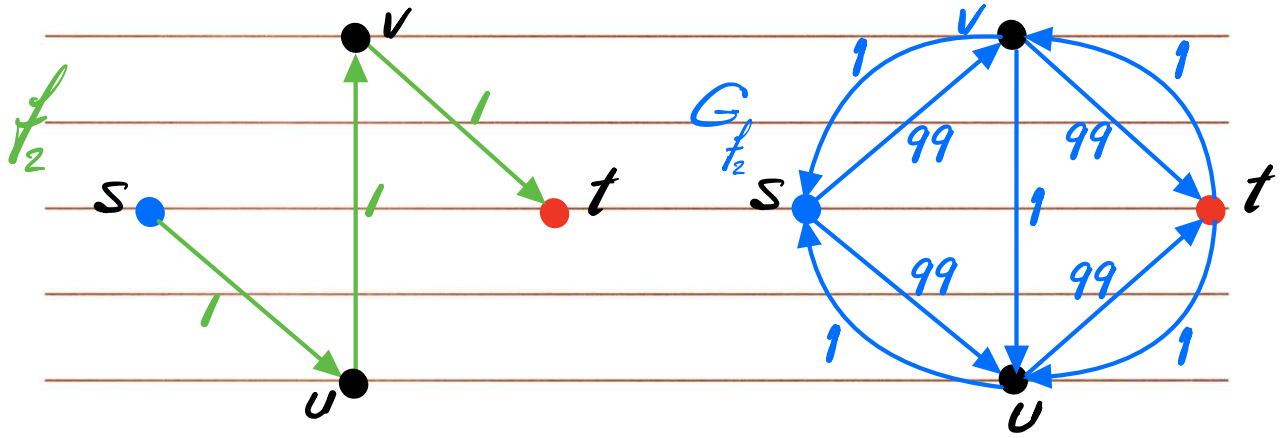
capacity

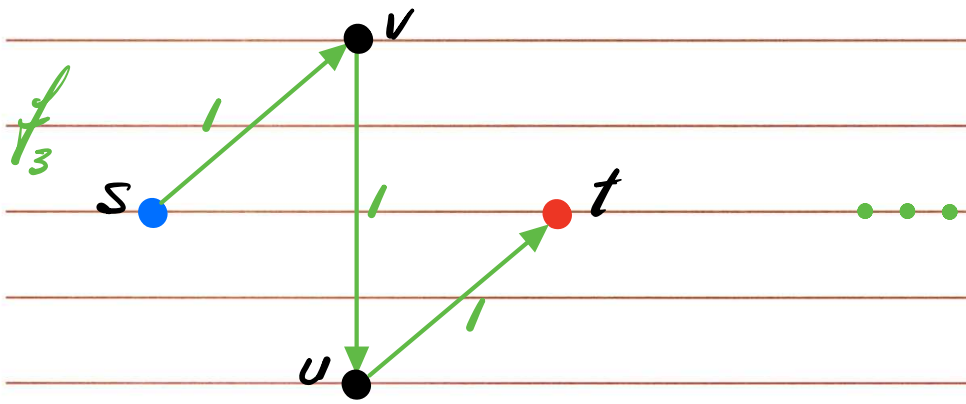
flow

B- In the worst case









Can take up to 200 iterations!

Scaled version of Ford-Fulkerson

Initially $f(e) = 0$ for all e in G
Set Δ to be the largest power of 2 that is no larger than the max. capacity of an edge out of s.
While $\Delta \geq 1$
  While there is an s-t path in $G_f(\Delta)$
    $O(m)$ Let P be a simple s-t path in $G_f(\Delta)$
    $O(m)$ $f' = \text{Augment}(f, c, P)$
    $O(m)$ $f = f'$
    $O(m)$ update $G_f(\Delta)$
  endwhile
  $\Delta = \Delta/2$
endwhile
Return f

$O(\log C)$?

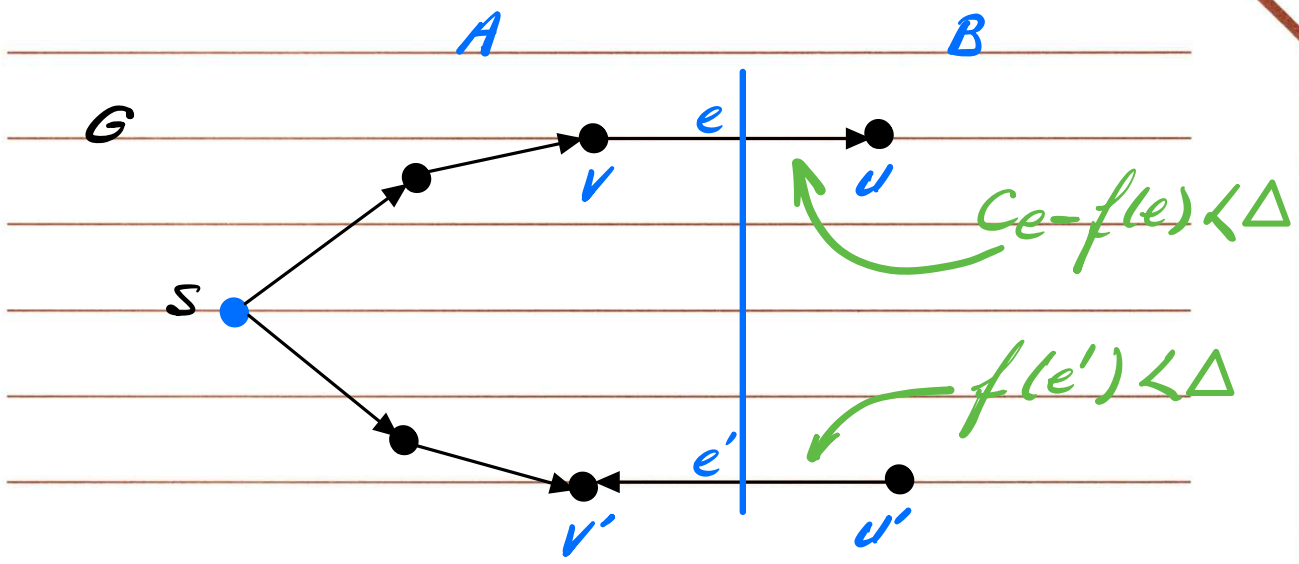
What is the max. no. of iterations in the inner while loop?

Fact: During the Δ-scaling phase, each augmentation increases the flow value by at least Δ.

Let f be the flow at the end of the Δ-scaling phase.

Claim: There is an s-t cut (A, B) in G for which $c(A, B) \leq v(f) + m\Delta$

Proof: Create sets A and B such that A includes all nodes v where there is an s-v path in $G_f(\Delta)$. Then $B = V - A$



Since there are at most m edges crossing the cut (A, B), each with residual cap. $< \Delta$, capacity of this cut $C(A,B) \leq v(f) + m\Delta$

So, if f^* is max. flow, it must be that

$$v(f^*) \leq C(A,B)$$

In other word, $v(f^*) \leq v(f) + m\Delta$

So, in the next scaling phase, where $\Delta' = \Delta/2$, how many iterations can we have at most? $m\Delta / (\Delta/2) = 2m$

While $\Delta \geq 1$

  While there is an s-t path in $G_f(\Delta)$

  $O(m)$ Let P be a simple s-t path in $G_f(\Delta)$

  $O(m)$ $f' = \text{Augment}(f, c, P)$

  $O(m)$ $f = f'$

  $O(m)$ update $G_f(\Delta)$

  endwhile

  $\Delta = \Delta/2$

endwhile

Return f

$O(m)$

$O(\log C)$

Overall complexity $= O(m^2 \log C)$

Is this an efficient solution? Yes!

More specifically, this is a weakly polyn. time solution.

Strongly vs Weakly polynomial time

An algorithm runs in strongly polynomial time if the no. of operations is bounded by a polynomial in the no. of integers in the input.

An algorithm runs in weakly polynomial time if the no. of operations is bounded by a polynomial in the no. of bits in the input (but not in the no. of integers in the input.)

Edmonds-Karp Algorithm

Same as Ford-Fulkerson, except that each augmentation path must be a shortest path with available capacity.

Can be shown to have running time of $O(nm^2)$

Strongly polyn.

Ford-Fulkerson	$O(Cm)$ pseudo-polyn.
Scaled version of F.F.	$O(m^2 \lg C)$ weakly polyn.
Edmonds-Karp	$O(nm^2)$ strongly polyn.
Orlin+KTR	$O(nm)$ strongly polyn.

More recently developed approximation methods solve max. flow in close to linear time WRT m. $\sim O(m^{4/3})$

1. You have successfully computed a maximum s-t flow f for a network G = (V; E) with 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 O(|V| + |E|) time.

2. You need to transport iron-ore from the mine to the factory. We would like to determine how long it takes to transport. For this problem, you are given a graph representing the road network of cities, with a list of k of its vertices ($t_1, t_2, \ldots, t_k$) which are designated as factories, and one vertex S (the iron-ore mine) where all the ore is present.

We are also given the following:

- Road Capacities (amount of iron that can be transported per minute) for each road (edges) between the cities (vertices).
- Factory Capacities (amount of iron that can be received per minute) for each factory (at $t_1, t_2, \ldots, t_k$)
- The amount of ore to be transported from the mine, C

Give a polynomial-time algorithm to determine the minimum amount of time necessary to transport and receive all the iron-ore at factories.

3. 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.

More formally, we can think of the USC campus as a graph, in which the nodes are locations, and edges are pathways or corridors. One of the nodes (the instructor's office) is the burglar's starting point, and several nodes (the USC gates) are the escape points — if the burglar reaches any one of those, the candy bars will be gone forever. Students and staff can be placed to monitor the edges. As it is hard to hide that many candy bars, the burglar cannot pass by a monitored edge undetected.

Give an algorithm to compute the minimum number of students/staff needed to ensure that the burglar cannot reach any escape points undetected (you don't need to output the corresponding assignment for students — the number is enough). As input, the algorithm takes the graph $G = (V,E)$ representing the USC campus, the starting point s, and a set of escape points $P \subseteq V$. Prove that your algorithm is correct and runs in polynomial time.

4. We define a most vital edge of a network as an edge whose deletion causes the largest decrease in the maximum s-t-flow value. Let f be an arbitrary maximum s-t-flow. Either prove the following claims or show through counterexamples that they are false:

(a) A most vital edge is an edge e with the maximum value of c(e).

(b) A most vital edge is an edge e with the maximum value of f (e).

(c) A most vital edge is an edge e with the maximum value of f (e) among edges belonging to some minimum cut.

(d) An edge that does not belong to any minimum cut cannot be a most vital edge.

(e) A network can contain only one most vital edge.