

Network Flow

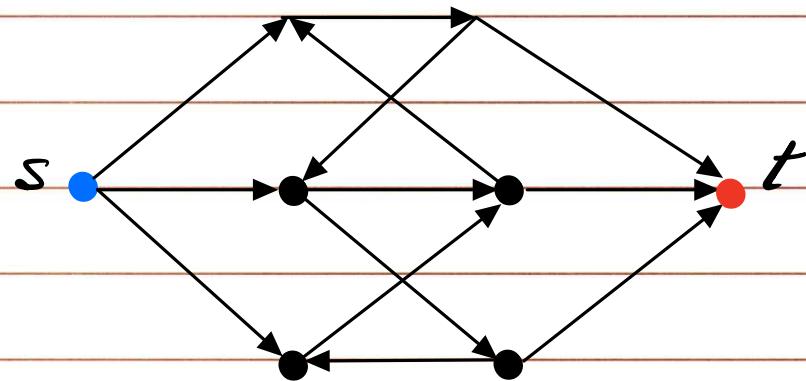
Examples of network flow problems:

- 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 ?

We will start with simpler networks.

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

- Each edge $e \in E$ 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
- 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:

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

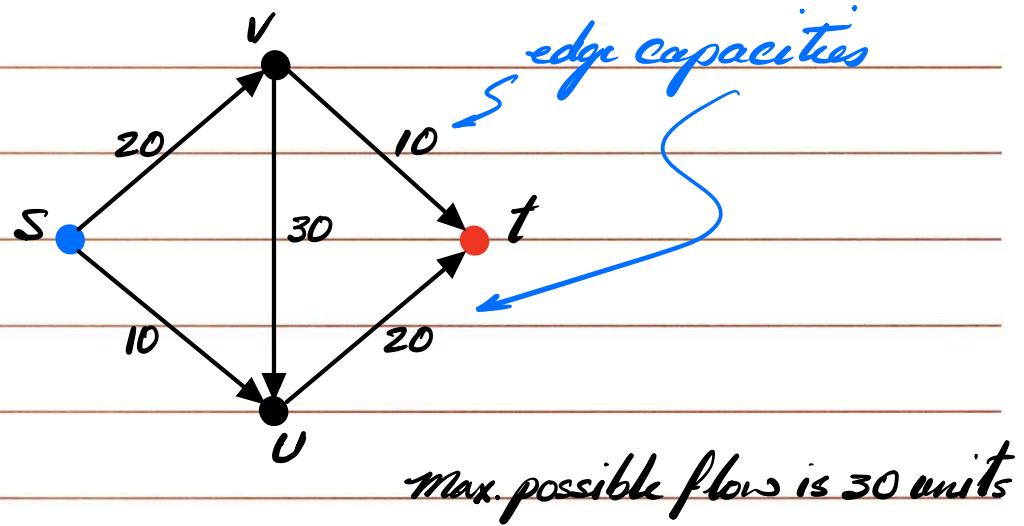
Our focus is on steady state flow, i.e no transient or burst flows.

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

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

Max Flow Problem Statement

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



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

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

To create G_f :

- If $f(e) = 0$

- If $f(e) = c_e$

- If $f(e) < c_e$

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 $b = \text{bottleneck}(P)$

for each edge $(V, U) \in P$

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

bottleneck(P) \leq

B- If e is a backward edge, Then

bottleneck(P) \leq

2- Check conservation of flow

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

$\text{MaxFlow}(G, s, t, C)$

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

$f' = \text{Augment}(f, C, P)$

$f = f'$

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

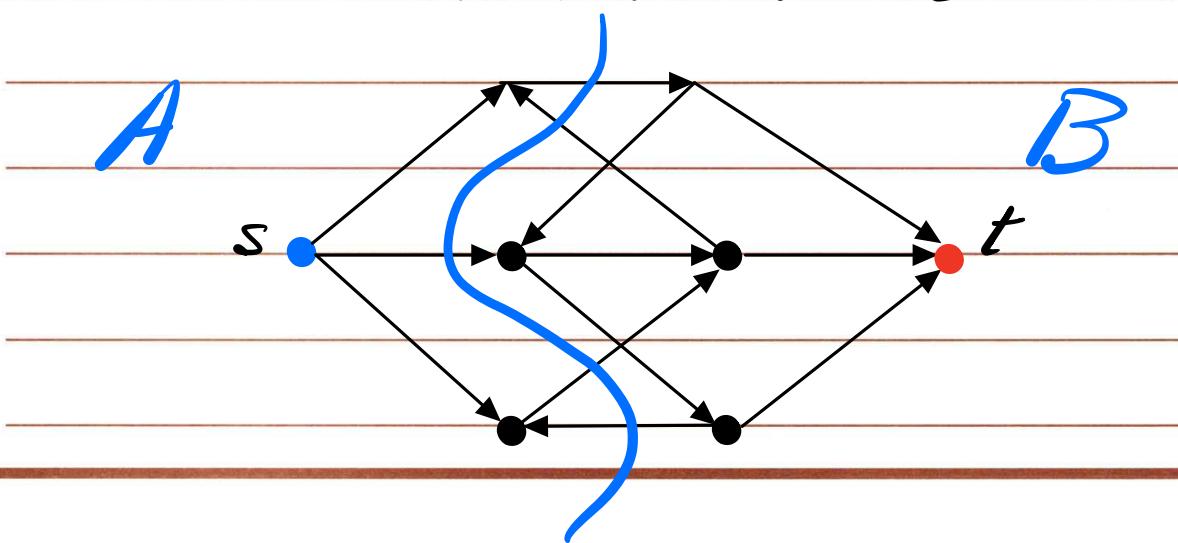
a- f is an integer flow

Proof by math. induction

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

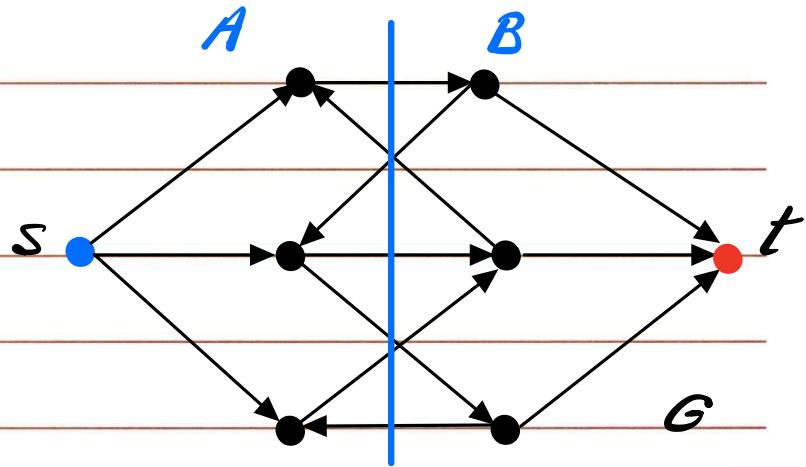
2- f is a Max Flow

Def. A cut divides nodes in the graph into \leq 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 .

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



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

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

Fact:

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

Complexity Analysis:

$\text{MaxFlow}(G, s, t, C)$

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

$f' = \text{Augment}(f, C, P)$

$f = f'$

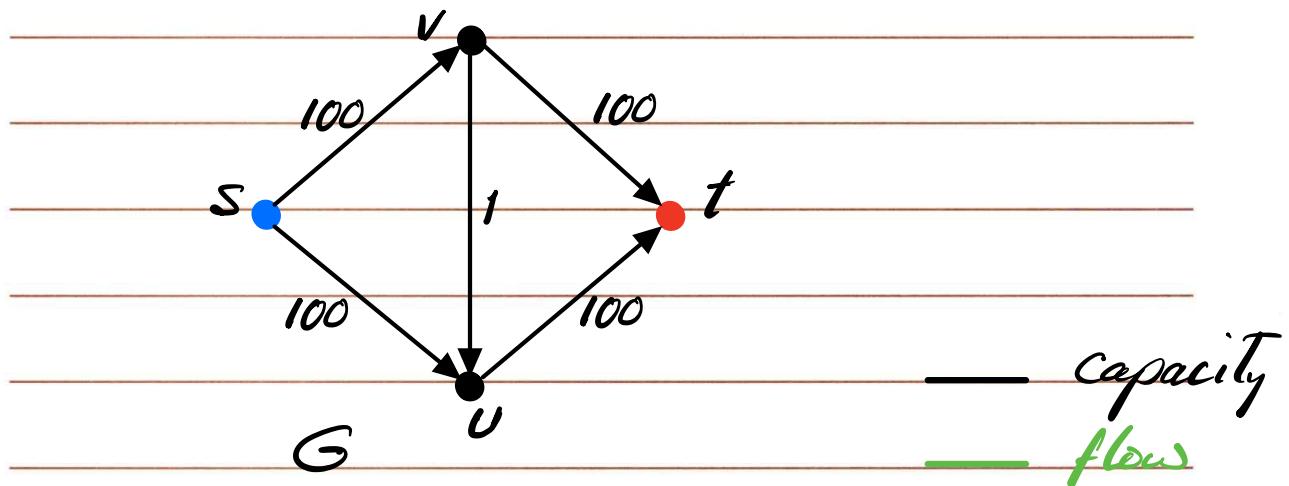
update G_f

endwhile

Return f

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

A- In the best case :



B- In the worst case

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 $G_f(\Delta)$

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

$f' = \text{Augment}(f, C, P)$

$f = f'$

update $G_f(\Delta)$

endwhile

$\Delta = \Delta/2$

endwhile

Return f

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 $B = V - A$

While $\Delta \geq 1$

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

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

$f' = \text{Augment}(f, C, P)$

$f = f'$

update $G_f(\Delta)$

endwhile

$\Delta = \Delta/2$

endwhile

Return f

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 $O(nm^2)$

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, \dots, 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, \dots, 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.
