CSCE 411: Design and Analysis of Algorithms

Lecture 17-18: Maximum s-t Flow

Date: Week 10 *Nate Veldt, updated by Samson Zhou*

Course Logistics

- Reading from Chapter 26 this week
- Homework 7 has been posted, due on Friday
- Test 2 next Thursday; Review next Tuesday

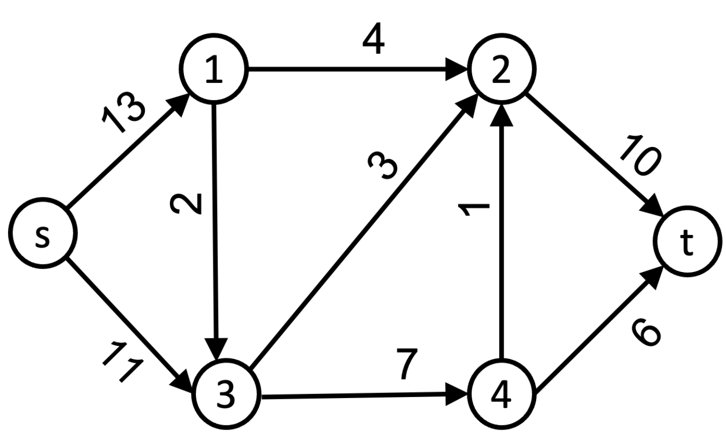
1 The Maximum s-t Flow Problem

Input to the Maximum s-t Flow Problem

- A weighted and directed graph $G = (V, E, w)$
- A source node s
- A sink node t

Goal: Route as much "flow" through the graph from s to t as possible, such that:

- The flow on an edge is bounded by
- The flow into a node (except for s and t) is equal to



One interpretation/application: transporting products/merchandise as efficiently as possible through a transportation network.

1.1 Defining s-t flows more formally

Given a weighted graph $G = (V, E, w)$, each $(u, v) \in E$ has a weight or *capacity* $w(u, v) = c(u, v)$.

A *flow* on G is a function

$$f : E \to \mathbb{R} \tag{1}$$

that satisfies two properties:

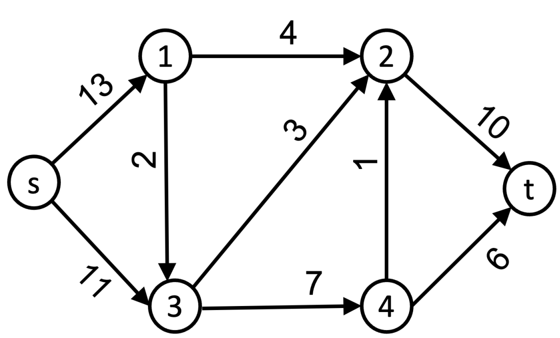
1. **Capacity constraints**: for each edge $(u, v) \in E$:

2. **Flow constraints**: for each node $v \notin \{s, t\}$

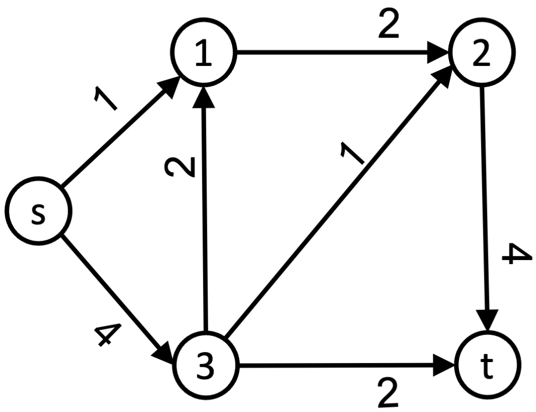
The *value* of a flow f is given by

$$|f| = \sum_{j\colon (s,j)\in E} f(s, j) - \sum_{u\colon (u,s)\in E} f(u, s) \tag{2}$$

Formal goal: find the flow function f^* with maximum value $|f^*|$.



Question 1. *What is the value of the flow f below?*



A *4*

B *5*

C *10*

D *15*

Question 2. *Is it a maximum flow?*

A *Yes it is*

B *No it is not*

C *It depends*

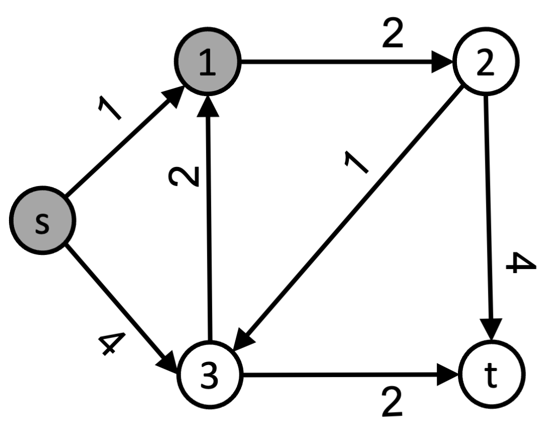
1.2 The minimum s-t cut problem

The minimum s-t cut problem takes the same type of input as the maximum s-t flow: a weighted directed graph $G = (V, E, w)$ with s and t nodes.

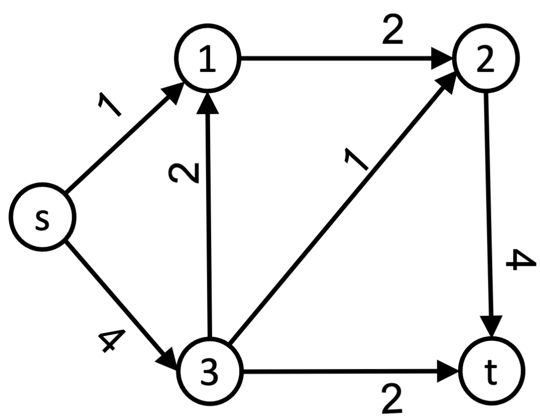
An *s-t cut* set is a set of nodes $S \subseteq V$ such that

The *value* of the cut is the weight of edges that cross from S to $V - S$. Formally:

What is the cut value below, where S is the set of gray nodes?



Question 3. *What is the value of the flow f below?*



A 1

B 2

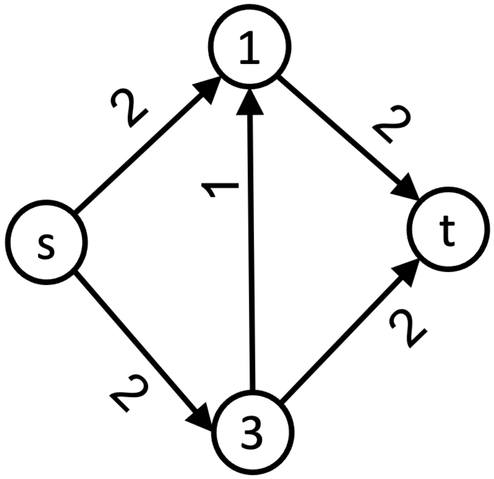
C 6

D 8

1.3 Relating cuts and flows

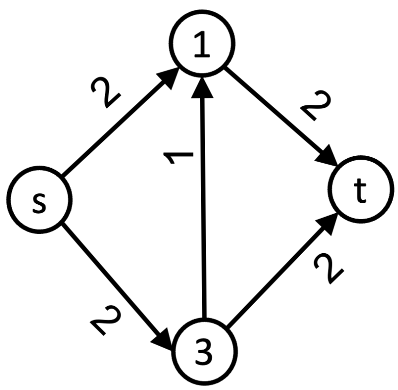
Lemma 1.1. *Let $G = (V, E, w)$ be a weighted directed graph. Let $S \subseteq V$ be any set with S be an s-t cut set, and let f be a flow. Then*

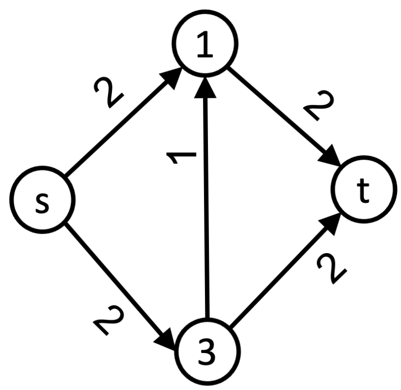
Consider the following graph. Find the optimal s-t flow value and then prove that it is optimal.



2 Finding maximum s-t flow

First idea. Repeatedly find paths from s to t, and keep adding flow until there are no more s-t paths.





How do we correct this? Let's try to keep track of flow that we could "undo".

2.1 The Residual Graph

Given a flow f, for a pair of nodes $(u, v) \in V \times V$, the *residual capacity* for (u, v) is

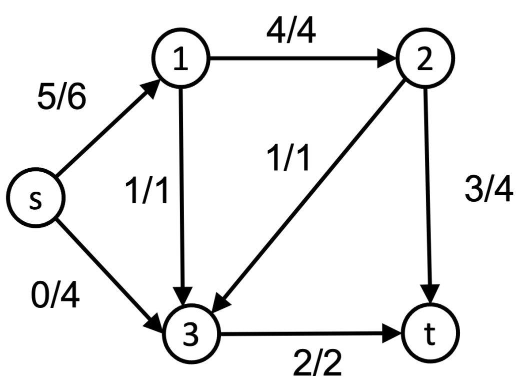
Informally, this is the amount of "space" left on the edge $c(u, v)$, plus the amount of flow from v to u that we could "undo".

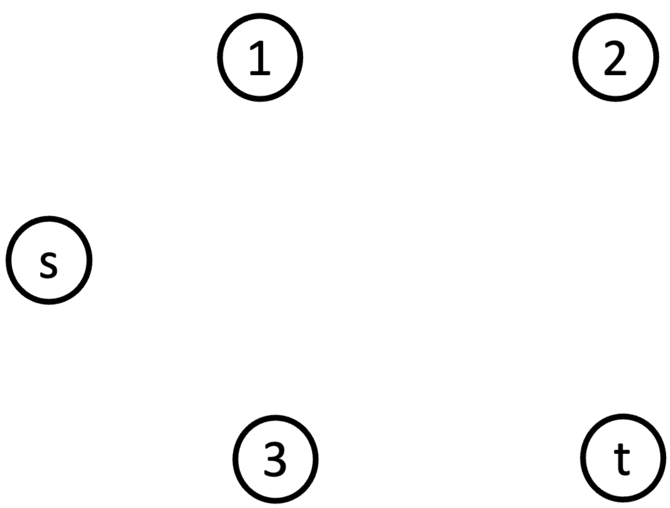
Given a flow f for a graph $G = (V, E, w)$, the *residual graph* $G_f = (V, E_f)$ is the graph where the edge set

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$$

This graph shows us where we can send more flow to improve on the flow f.

Activity: draw the residual graph for the following flow





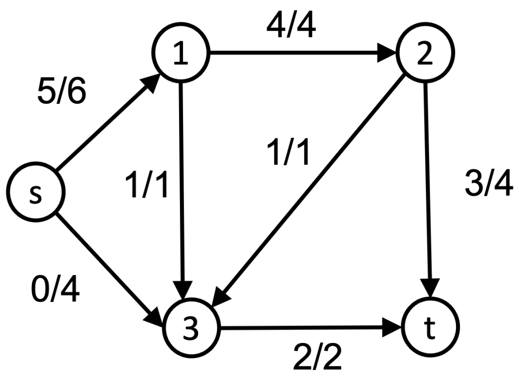
2.2 Augmenting Flows and Paths

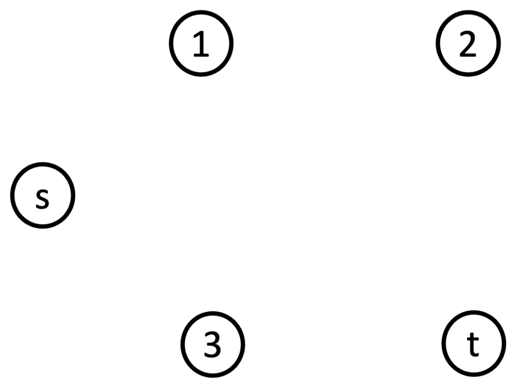
Let f be an s-t flow in $G = (V, E)$ and f' be a flow in the residual graph $G_f = (V, E_f)$. Then we define the *augmentation* of f by f' as:

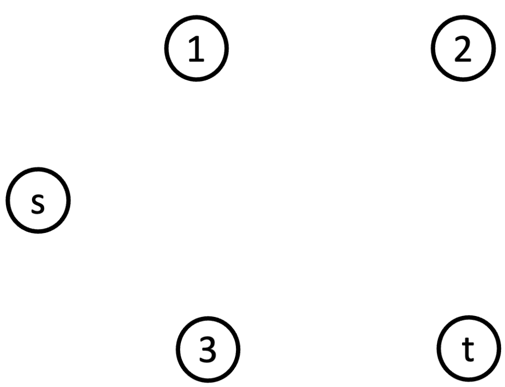
$$f \uparrow f' = f(u, v) + f'(u, v) - f'(v, u) \tag{3}$$

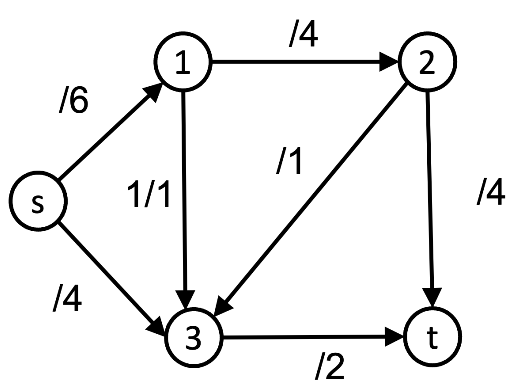
Lemma 2.1. *The function $f \uparrow f'$ is a valid flow in G, and it has flow value $|f| + |f'|$.*

Proof: a whole bunch of bookkeeping. We will skip this. But we can illustrate it below.









An *augmenting path* p is a simple path (simple = no cycles) from s to t in the residual network G_f.

The *residual capacity* of this path p is the maximum amount we can send on p:

Sending $c_f(p)$ flow along every edge in this path gives us a flow f_p in G_f that we can add to f to improve it.

Theorem 2.2. *(Max-flow Min-cut Theorem) Let f be an s-t flow on some graph $G = (V, E)$. The following three conditions are equivalent:*

1. *f is a maximum s-t flow*
2. *There are no augmenting paths in the residual graph G_f*
3.

3 The Basic Ford-Fulkerson Algorithm

Idea: f is a max-flow if and only if there are no augmenting paths. So let's just keep finding augmenting paths until we're done!

The Ford-Fulkerson algorithm will always maintain the invariant that for any pair (u, v), at most one of $\{f(u, v), f(v, u)\}$ will be greater than zero.

FordFulkersonBasic(G, s, t)

```
for (u,v) ∈ E do
    f(u,v) = 0
end for
while there exists an s-t path p in G_f do
    c_f(p) = min{c_f(u,v): (u,v) is in p}
    for (u,v) ∈ p do
        m = min{c_f(p), f(v,u)}
        ℓ = c_f(p) − m
        f(v,u) ← f(v,u) − m
        f(u,v) ← f(u,v) + ℓ
    end for
end while
```

For $(u, v) \in p$, we first use any of the flow $c_f(p)$ to undo flow previously sent on (v, u).

Then, if any of $c_f(p)$ remains, we send it along (u, v).

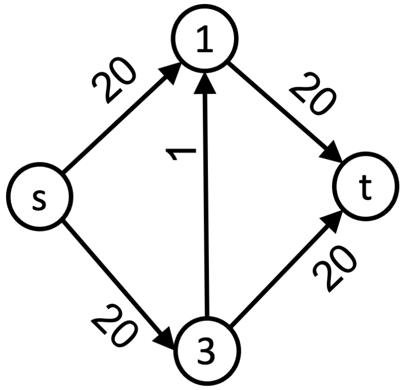
3.1 Runtime Analysis

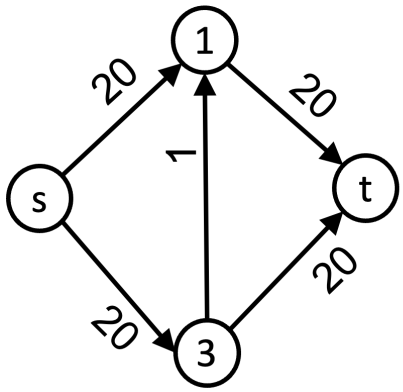
- f^* is the maximum flow and $|f^*|$ the maximum flow value.
- Assume all weights are integers.
- Let f be the flow we are growing as the algorithm progresses.

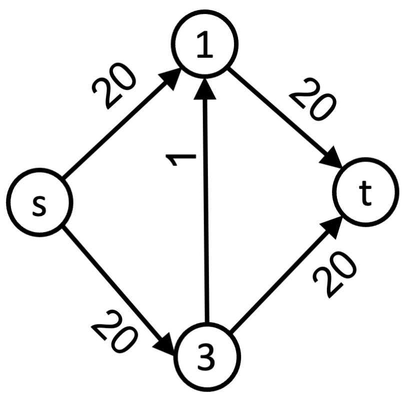
Answer the following questions:

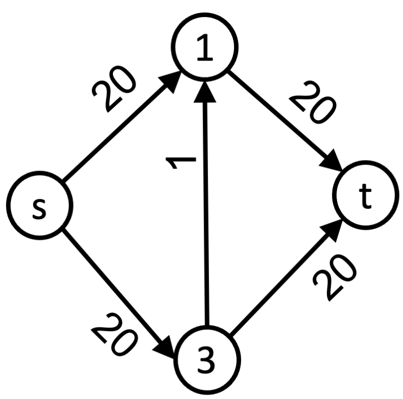
1. What is the runtime complexity for finding an s-t path p in G_f?
2. What is the minimum amount by which we can increase f in each iteration?
3. What is the maximum number of paths we might have to find before we are done?
4. What is an overall runtime bound for FordFulkersonBasic?

3.2 How bad can the runtime be in practice?









4 The Edmonds-Karp Algorithm

The Edmonds-Karp Algorithm is a variation on Ford-Fulkerson that chooses an augmenting path p by finding the directed path from s to t with the smallest number of edges.

Question 4. *Which algorithm should we use as a subroutine for finding paths for Edmonds-Karp?*

A *Breadth first search*

B *Depth first search*

C *Topological sort*

D *Single source shortest path problem*

E *Hm...not sure.*

4.1 Shortest path distances increases monotonically

Let f be an s-t flow for input $G = (V, E, s, t)$ and G_f be the residual graph. Define

$$\delta_f(s, v) = \text{ the shortest unweighted path distance from } s \text{ to } v \text{ in } G_f$$

Lemma 4.1. *For every $v \in V$, the distance $\delta_f(s, v)$ increases monotonically with each flow augmentation.*

Translation: as we keep finding augmenting paths p and sending more flow f_p to f, the distance between s and every node either stays the same, or increases.

Theorem 4.2. *The total number of flow augmentation steps performed by Edmonds-Karp is $O(VE)$.*

Proof.

- Let p be an augmenting path in G_f.

- An edge $(u, v) \in p$ is *critical* if $c_f(p) = c_f(u, v)$, meaning it is the smallest capacity edge in that path.

- When we push $c_f(p)$ flow through p, the edge (u, v) disappears from G_f

- At least one edge on each path p is critical.

- Claim: Each of the $|E|$ edges can be critical at most $|V|/2$ times.

Proving the claim: (u,v) becomes critical at most $|V|/2$ times.

- Let u and v be nodes in some edge in E.

- When (u, v) is critical for the first time, $\delta_f(s, v) = \delta_f(s, u) + 1$

  Because they are on a shortest path

- Then (u, v) disappears from the residual graph, and can only re-appear after (v, u) is on some future augmenting path. Say that (v, u) is on an augmenting path when the new flow on G is f', then

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1.$$

- We know that $\delta_f(s, v) \leq \delta_{f'}(s, v)$

- So we have

$$\delta_{f'}(s, u) =$$

- From the first to the second time (u, v) becomes critical, the distance from s to u increases by at least 2.

- If (u, v) becomes critical more than $|V|/2$ times, then the distance from s to u increases by more than $2|V|/2 \geq |V|$.

- Thus, (u, v) becomes critical at most $|V|/2 = O(V)$ times.

$\square$