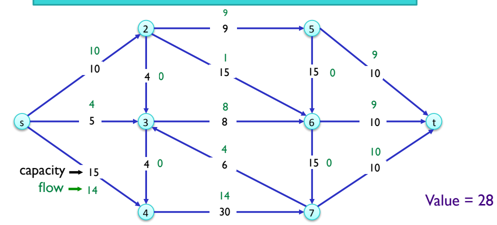
Objectives

- Network Flow
  - Motivation
  - Max flow
  - Min cut

Review: Maximum Flow Problem

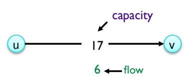
- Make network most efficient
  - Use most of available capacity

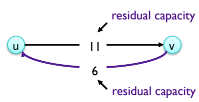
Goal: Find *s-t* flow of maximum value



Review: Residual Graph: G_f

- Original edge: e = (u, v) ∈ E
  - Flow f(e), capacity c(e)
- Residual edge
  - e = (u, v) w/ capacity c(e) - f(e)
  - e^R = (v, u) with capacity f(e)
    - To undo flow
- Residual graph: $G_f = (V, E_f)$
  - Residual edges with *positive* residual capacity
  - $E_f = \underbrace{\{e : f(e) < c(e)\}}_{\text{Forward edges}} \cup \underbrace{\{e^R : f(e) > 0\}}_{\text{Backward edges}}$





Augmenting Path Algorithm

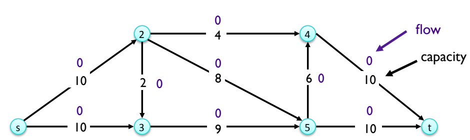
c=capacity

```
Ford-Fulkerson(G, s, t, c)
  foreach e ∈ E  f(e) = 0  # initially no flow
  Gf = residual graph

  while there exists augmenting path P
     f = Augment(f, c, P)      # change the flow
     update Gf             # build a new residual graph

  return f
```

```
Augment(f, c, P)
  b = bottleneck(P) # edge on P with least capacity
  foreach e ∈ P
     if (e ∈ E) f(e) = f(e) + b  # forward edge, ↑ flow
     else       f(eR) = f(e) - b  # forward edge, ↓ flow
  return f
```


Ford-Fulkerson Algorithm

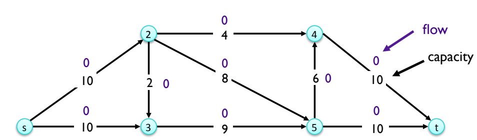
G:



Flow value = 0

Ford-Fulkerson Algorithm

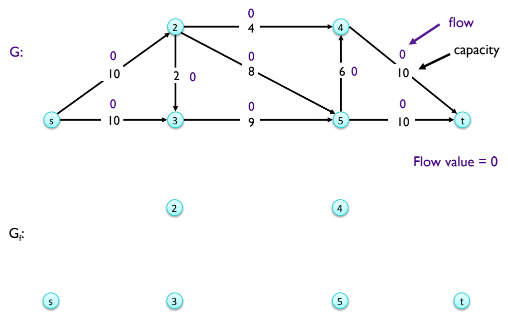
G:



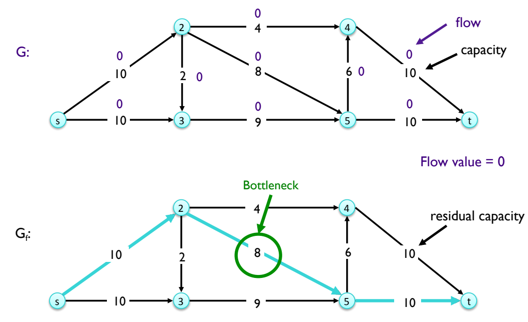
Flow value = 0

What does the residual graph look like?

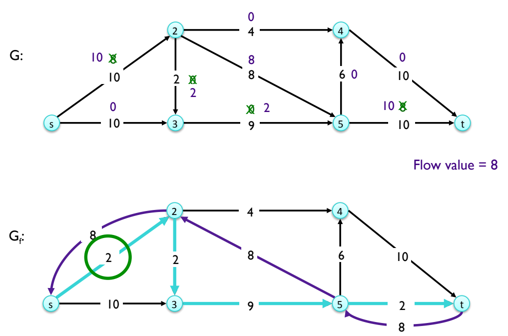
Ford-Fulkerson Algorithm



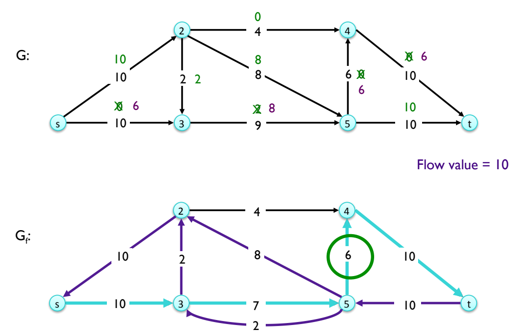
Ford-Fulkerson Algorithm



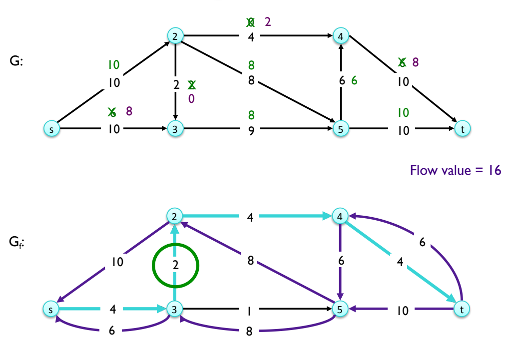
Ford-Fulkerson Algorithm



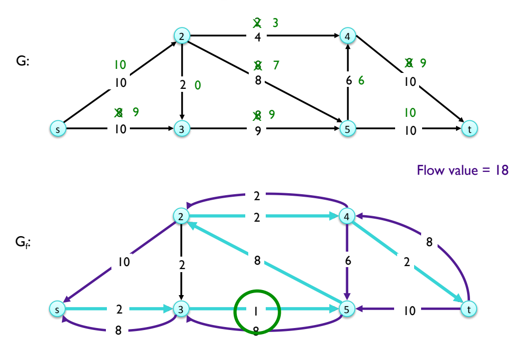
Ford-Fulkerson Algorithm



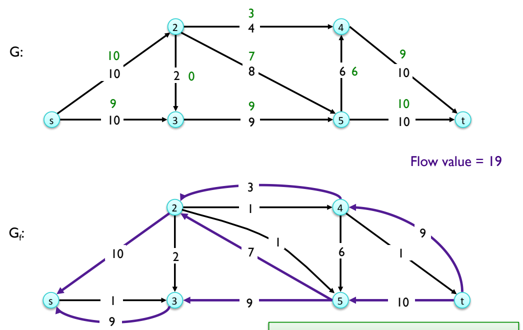
Ford-Fulkerson Algorithm



Ford-Fulkerson Algorithm

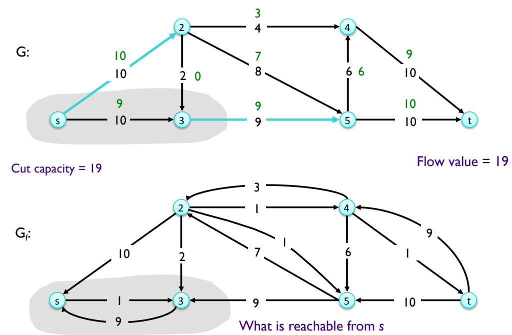


Ford-Fulkerson Algorithm



How do we know we're done?

Ford-Fulkerson Algorithm



Analyzing Augmenting Path Algorithm

```
Ford-Fulkerson(G, s, t, c)
  foreach e ∈ E  f(e) = 0  # initially no flow
  Gf = residual graph

  while there exists augmenting path P
     f = Augment(f, c, P)      # change the flow
     update Gf               # build a new residual graph

  return f
```

```
Augment(f, c, P)
  b = bottleneck(P) # edge on P with least capacity
  foreach e ∈ P
     if (e ∈ E) f(e) = f(e) + b  # forward edge, ↑ flow
     else       f(eR) = f(e) - b  # forward edge, ↓ flow
  return f
```

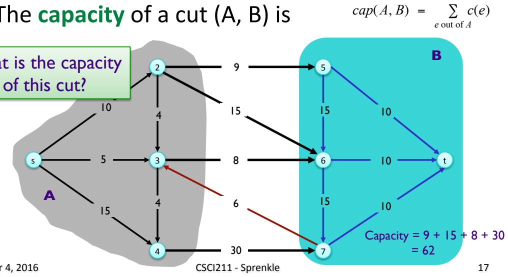
Why does alg work? What is happening at each iteration?
What is the running time?
Need more analysis …

MINIMUM CUTS

Cuts

- An **s-t cut** is a partition (A, B) of V with s ∈ A and t ∈ B
- The **capacity** of a cut (A, B) is $cap(A,B) = \sum_{e \text{ out of } A} c(e)$

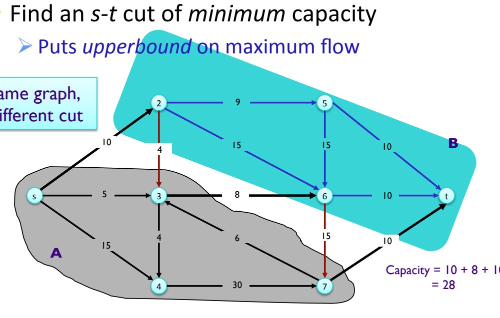
What is the capacity of this cut?



Minimum Cut Problem

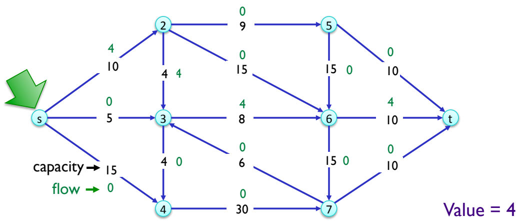
- Find an *s-t* cut of *minimum* capacity
  - Puts *upperbound* on maximum flow

Same graph, different cut



Recall

- The **value of a flow** f is $v(f) = \sum_{e \text{ out of } s} f(e)$

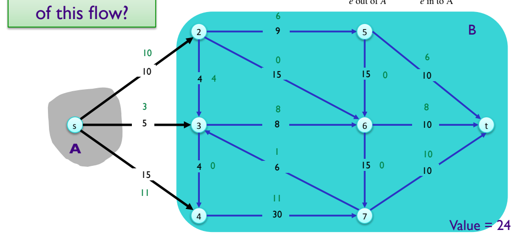


Flow Value Lemma

- Let *f* be *any* flow, and let (A, B) be *any s-t* cut. Then, the **value of the flow** is = $f^{out}(A) - f^{in}(A)$.

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

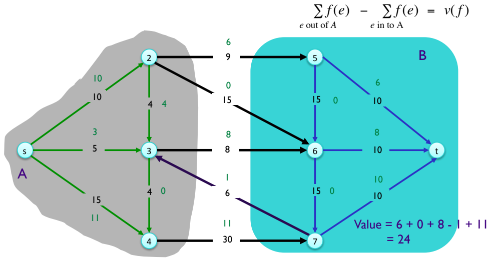
What is the value of this flow?



Flow Value Lemma

- Let *f* be *any* flow, and let (A, B) be *any s-t* cut. Then, the **value of the flow** is = $f^{out}(A) - f^{in}(A)$.

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

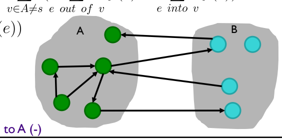


Flow Value Lemma (FVL)

- Let *f* be *any* flow, and let (A, B) be *any s-t* cut.
- Then
- Pf. $v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$

$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } s} f(e) \quad \text{By definition} \\
&= \sum_{e \text{ out of } s} f(e) + \sum_{v \in A \neq s} \Big(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \Big) \\
&= \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e) + \sum_{v \in A \neq s} \Big(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \Big) \\
&= \sum_{v \in A} \Big(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \Big) \\
&= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)
\end{aligned}
$$

by flow conservation, all terms except v = s are 0



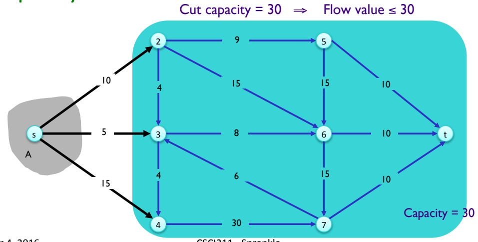
Possibilities for edge e:
- Both ends in A (0)
- Points out from A (+), Points in to A (-)

Weak Duality

- Let *f* be any flow and let (A, B) be any *s-t* cut.

➔ Then the value of the flow is *at most* the cut's capacity

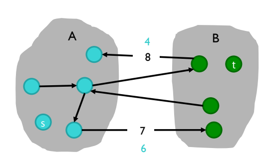
Cut capacity = 30 ⇒ Flow value ≤ 30



Weak Duality

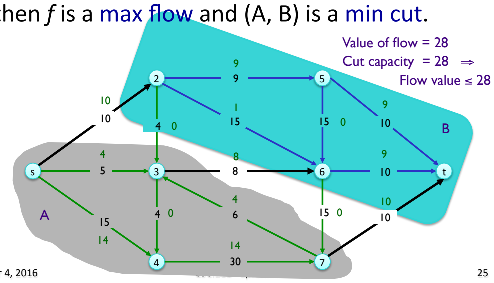
- Let *f* be any flow. Then, for any *s-t* cut (A, B) $v(f) \leq cap(A, B)$.
- Pf.

By FVL
$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&\leq \sum_{e \text{ out of } A} f(e) \\
&\leq \sum_{e \text{ out of } A} c(e) \\
&= cap(A, B)
\end{aligned}
$$



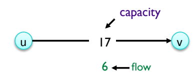
Certificate of Optimality

- Corollary. Let *f* be any flow, and let (A, B) be any cut. If v(f) = cap(A, B), then *f* is a max flow and (A, B) is a min cut.

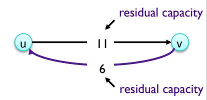


Recall: Residual Graph G_f

- Original edge: e = (u, v) ∈ E
  - Flow f(e), capacity c(e)



- Residual edge
  - e = (u, v) w/ capacity c(e) - f(e)
  - e^R = (v, u) with capacity f(e)
    - To undo flow



- Residual graph: $G_f = (V, E_f)$
  - Residual edges with *positive* residual capacity
  - $E_f = \underbrace{\{e : f(e) < c(e)\}}_{\text{Forward edges}} \cup \underbrace{\{e^R : f(e) > 0\}}_{\text{Backward edges}}$

Recall: Augmenting Path Algorithm

```
Ford-Fulkerson(G, s, t, c)
   foreach e ∈ E  f(e) = 0  # initially no flow
   Gf = residual graph

   while there exists augmenting path P
      f = Augment(f, c, P)      # change the flow
      update Gf             # build a new residual graph

   return f
```

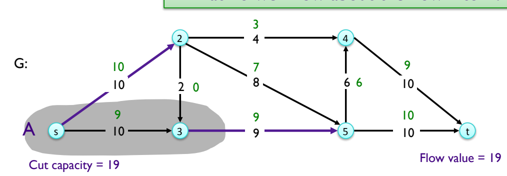
```
Augment(f, c, P)
   b = bottleneck(P) # edge on P with least capacity
   foreach e ∈ P
      if (e ∈ E) f(e) = f(e) + b  # forward edge, ↑ flow
      else       f(eR) = f(e) - b  # forward edge, ↓ flow
   return f
```

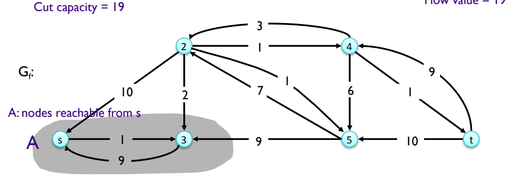

Intuition Behind Correctness of F-F Algorithm

- Let A be set of vertices *reachable* from *s* in residual graph at end of F-F alg execution
- By definition of A, s ∈ A
- By definition of the F-F algorithm's resulting flow, t ∉ A

Ford-Fulkerso

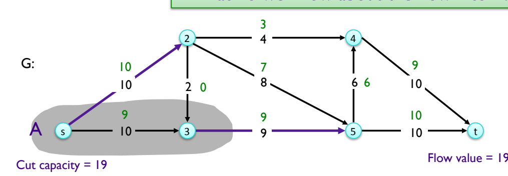
- What do we know about the flow out of A?
- What do we know about the flow into A?



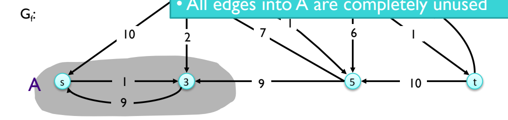


Ford-Fulkerso

- What do we know about the flow out of A?
- What do we know about the flow into A?



- All edges out of A are completely saturated
- All edges into A are completely unused



Max-Flow Min-Cut Theorem

Augmenting path theorem.
Flow *f* is a max flow iff there are no augmenting paths.

Max-flow min-cut theorem. [Ford-Fulkerson 1956]
The value of the max flow is equal to the value of the min cut.

- Proof strategy. Prove both simultaneously by showing the following are equivalent:
  - (i) There exists a cut (A, B) such that v(f) = cap(A, B).
  - (ii) Flow f is a max flow.
  - (iii) There is no augmenting path relative to f.

See formal proof in book

Analyzing Augmenting Path Algorithm

```
Ford-Fulkerson(G, s, t, c)
  foreach e ∈ E  f(e) = 0  # initially no flow
  G_f = residual graph

  while there exists augmenting path P
     f = Augment(f, c, P)      # change the flow
     update G_f            # build a new residual graph

  return f
```

Costs?

```
Augment(f, c, P)
  b = bottleneck(P) # edge on P with least capacity
  foreach e ∈ P
     if (e ∈ E) f(e) = f(e) + b  # forward edge, ↑ flow
     else       f(e^R) = f(e) - b  # forward edge, ↓ flow
  return f
```


Analyzing Augmenting Path Algorithm

```
       Ford-Fulkerson(G, s, t, c)
O(m)     foreach e ∈ E  f(e) = 0  # initially no flow
O(m)     G_f = residual graph
Find path: O(m); Iterations: O(F) iterations, where F = max flow
         while there exists augmenting path P
O(m)        f = Augment(f, c, P)      # change the flow
O(m)        update G_f            # build a new residual graph

         return f
```

Total: O(Fm)

```
       Augment(f, c, P)
O(n)     b = bottleneck(P) # edge on P with least capacity
O(n)     foreach e ∈ P
O(1)        if (e ∈ E) f(e) = f(e) + b  # forward edge, ↑ flow
O(1)        else       f(e^R) = f(e) - b  # forward edge, ↓ flow
         return f
```

Total: O(n) → O(m), since n ≤ 2m

Running Time

- Assumption. All capacities are integers between 1 and F.
- Invariant. Every flow value f(e) and every residual capacity's $c_f(e)$ remains an integer throughout algorithm.

- Theorem. Algorithm terminates in at most v(f*) ≤ nF iterations.
- Pf. Each augmentation increases value by at least 1.
- Corollary. If F = 1, Ford-Fulkerson runs in O(mn) time.

- Integrality theorem. If all capacities are integers, then there exists a max flow *f* for which every flow value f(e) is an integer.
- Pf. Since algorithm terminates, theorem follows from invariant.

Looking Ahead

- Wiki: Due tonight (7.1-7.2, 7.5, 7.7)
  - 7.5 won't be discussed in class

- Problem Set 9 due Friday