## Objectives

- Network Flow
  - Max flow
  - Min cut



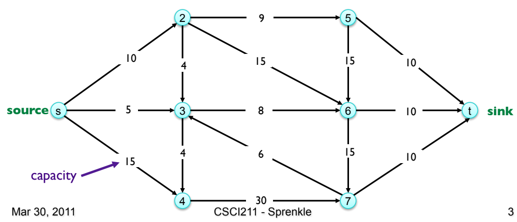
## Motivating Flow Network Problems

- Modeling *transportation* networks
  - Edges: carry traffic
  - Nodes: pass traffic between edges
- Can represent many different types of problems
  - Instead of looking at all possibilities, formulate as a flow problem



## Flow Network

- G = (V, E) = directed graph, no parallel edges
- Two distinguished nodes: *s* = source, *t* = sink
- c(e) = capacity of edge e, > 0



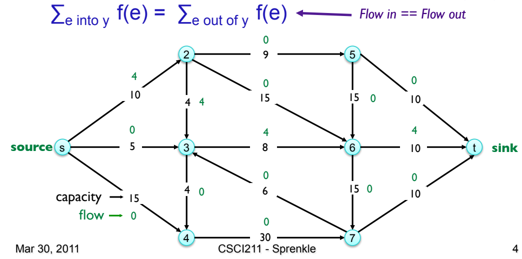



## Flows

- An **s-t flow** is a function that satisfies
  - ***Capacity* condition**: For each e ∈ E: $0 \le f(e) \le c(e)$
  - ***Conservation* condition**: For each v ∈ V – {s, t}: $\sum_{e \text{ into } y} f(e) = \sum_{e \text{ out of } y} f(e)$

*Flow can't exceed capacity*

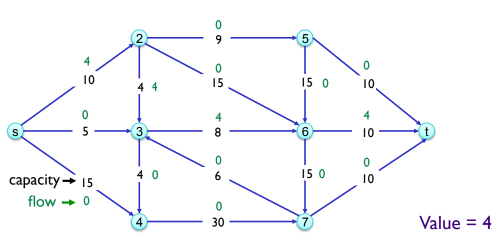
*Flow in == Flow out*






## Flows

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

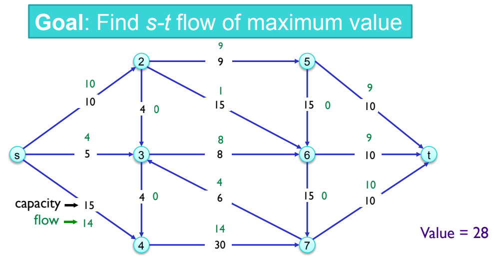





## Maximum Flow Problem

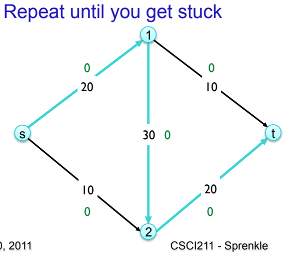
- Make network most efficient
  - Use most of available capacity

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



## Towards a Max Flow Algorithm

- Greedy algorithm
  - Start all edges e ∈ E at f(e) = 0
  - Find an *s-t* path P with the most capacity: f(e) < c(e)
  - Augment flow along path P
  - Repeat until you get stuck

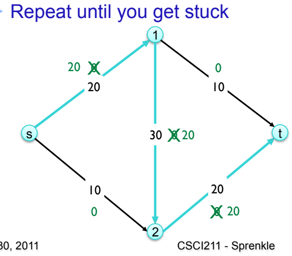



Flow value = 0



## Towards a Max Flow Algorithm

- Greedy algorithm
  - Start all edges e ∈ E at f(e) = 0
  - Find an *s-t* path P with the most capacity: f(e) < c(e)
  - Augment flow along path P
  - Repeat until you get stuck




Is this optimal?

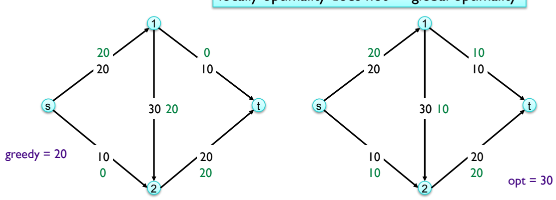
Flow value = 20



## Towards a Max Flow Algorithm

- Greedy algorithm
  - Start all edges e ∈ E at f(e) = 0
  - Find an *s-t* path P with the most capacity: f(e) < c(e)
  - Augment flow along path P
  - Repeat until you get stuck

locally optimality does not ⇒ global optimality



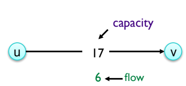



## RESIDUAL GRAPHS



## Towards a Residual Graph

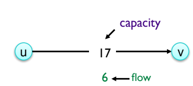
- Original edge: e = (u, v) ∈ E
  - Flow f(e), capacity c(e)

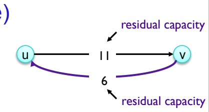





## Towards a Residual Graph

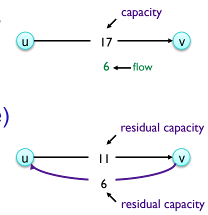
- Original edge: e = (u, v) ∈ E
  - Flow f(e), capacity c(e)
- Residual edge
  - e = (u, v) w/ capacity c(e) - f(e)
  - $e^R$ = (v, u) with capacity f(e)
    - To undo flow



## Residual Graph: $G_f$

- 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
  - $E_f = \underbrace{\{e : f(e) < c(e)\}}_{\text{Forward edges}} \cup \underbrace{\{e^R : f(e) > 0\}}_{\text{Backward edges}}$






## Applying Residual Graph

- Used to find the maximum flow
  - Use similar idea to greedy algorithm
- Residual path: simple *s-t* path in $G_f$
  - Also known as *augmenting path*



## Augmenting Path Algorithm

c=capacity

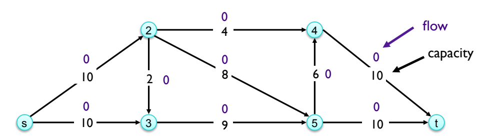
```
Ford-Fulkerson(G, s, t, c)
   foreach e ∈ E  f(e) = 0  # initially no flow
   Gf = residual graph

   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:

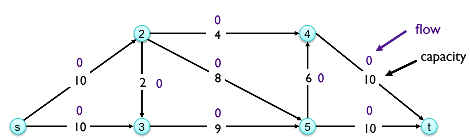



Flow value = 0



## Ford-Fulkerson Algorithm

G:



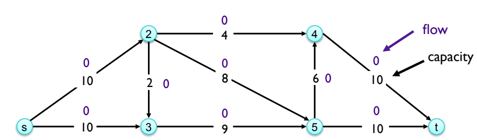

Flow value = 0

What does the residual graph look like?



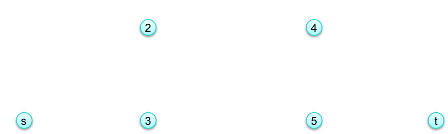
## Ford-Fulkerson Algorithm

G:

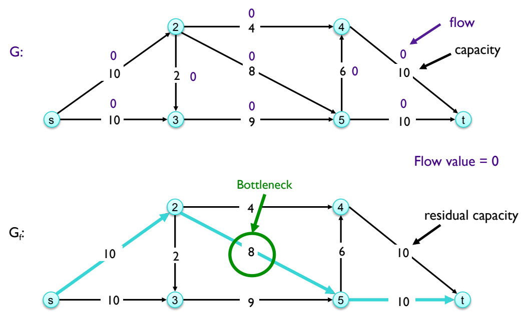



Flow value = 0

$G_f$:

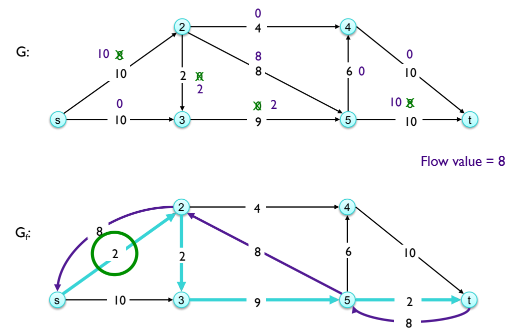


## Ford-Fulkerson Algorithm



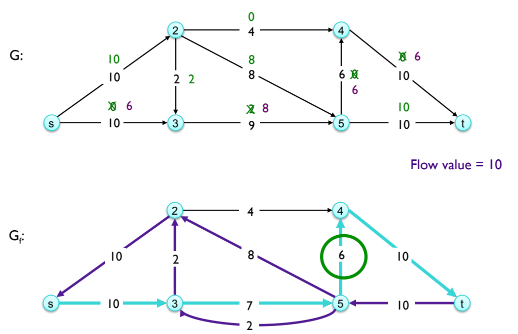



## Ford-Fulkerson Algorithm



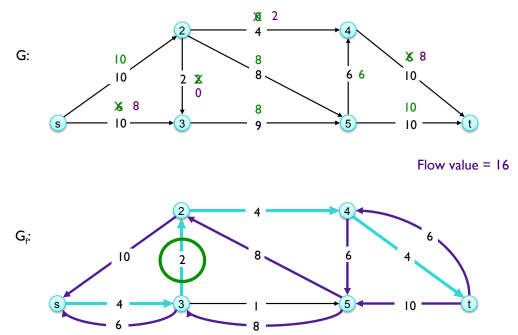



## Ford-Fulkerson Algorithm



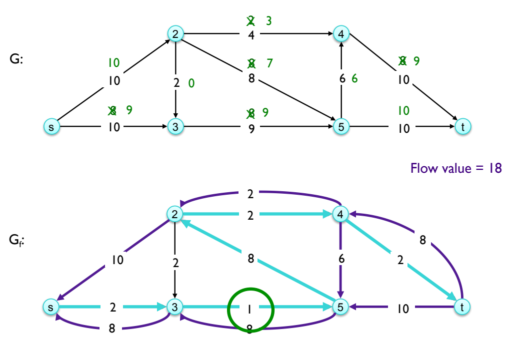



## Ford-Fulkerson Algorithm



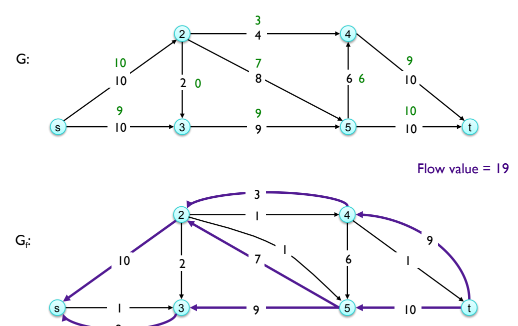



## Ford-Fulkerson Algorithm



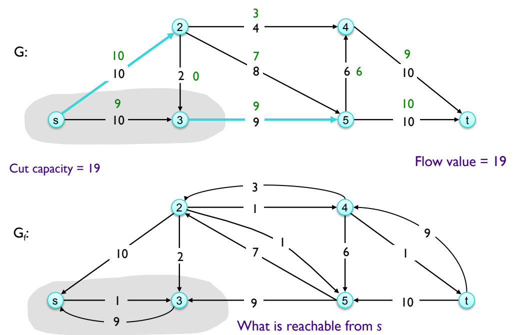



## Ford-Fulkerson Algorithm




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 …



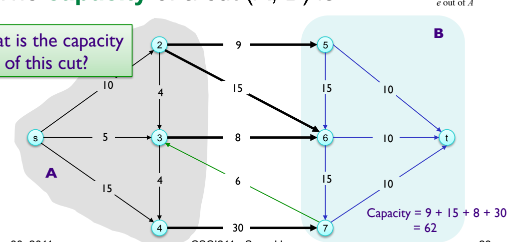
# MINIMUM CUTS



## 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?

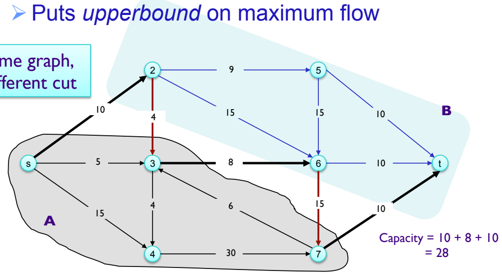





## Minimum Cut Problem

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

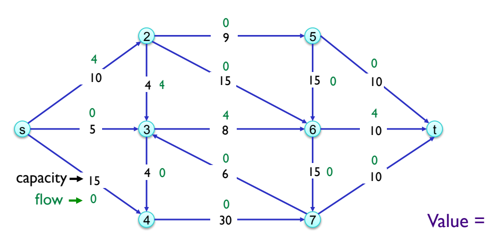
Same graph, different cut






## Recall

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

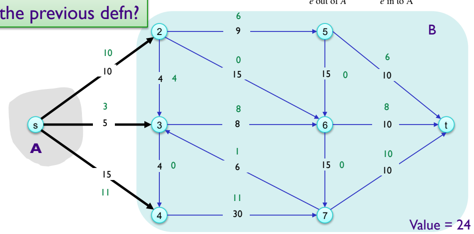


## Flow Value Lemma

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

How does this relate to the previous defn?

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

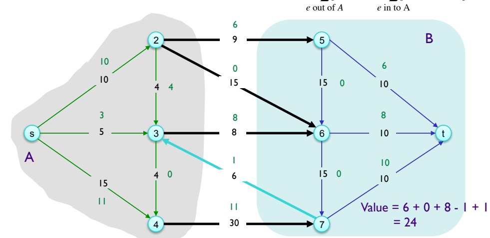





## Flow Value Lemma

- 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

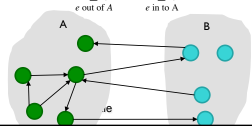
- Let *f* be *any* flow, and let (A, B) be *any s-t* cut.
- Then $\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$.
- Pf.

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

by flow conservation, all terms except v = s are 0
$$= \sum_{v \in A} \left( \sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right)$$

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

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

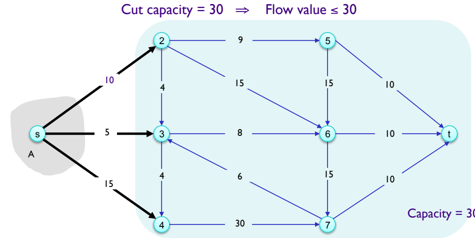





## Weak Duality

- 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

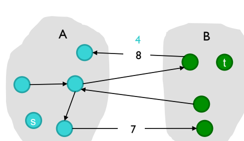





## Weak Duality

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

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

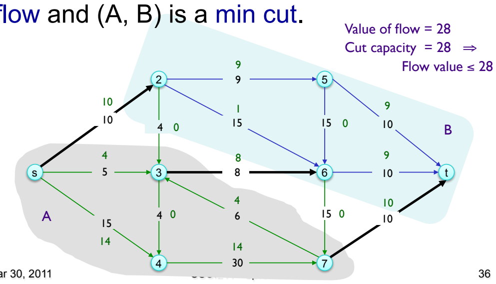





## Certificate of Optimality

- 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.

Value of flow = 28
Cut capacity = 28 ⇒ Flow value ≤ 28



## This Week

- Problem Set 8 due Friday
  - Implementing pretty printing
- Start reading chapter 7