Objectives

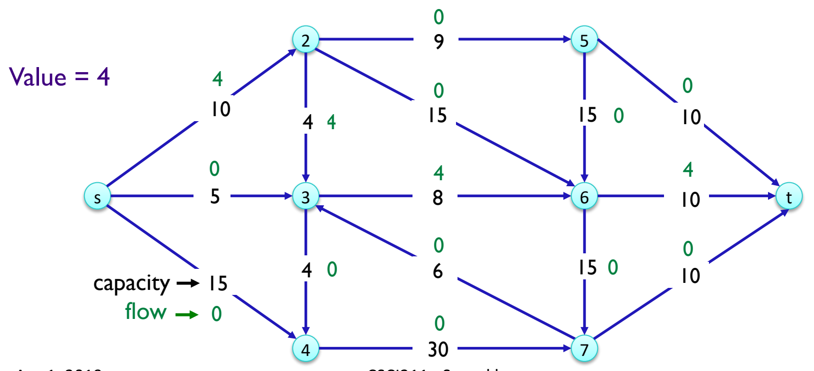
- Network Flow
  - Circulation
  - Application: Survey Design
  - Application: Airline Scheduling

Review

- What is a flow network?
- What is our usual goal given a flow network?
- What is the Ford-Fulkerson algorithm?
  - What is its purpose?
  - Why does it work?
- What is the min-cut?
  - How does it relate to the max flow?

Review: Network Flows

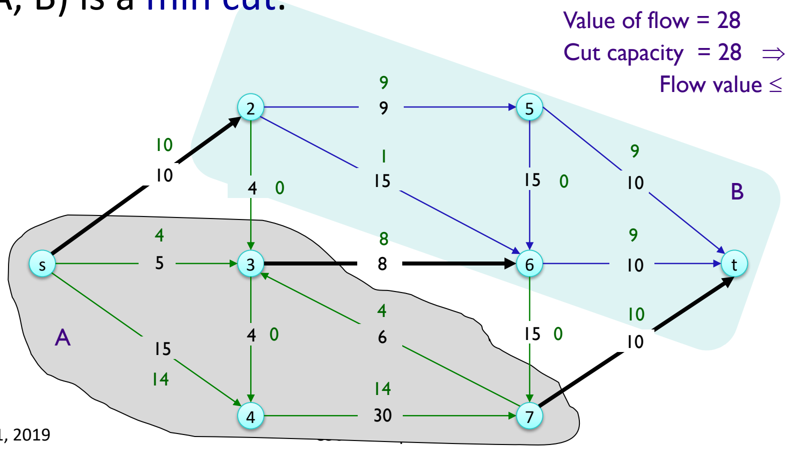
- An **s-t flow** is a function that satisfies
  - ***Capacity* condition**: For each $e \in E$: $0 \le f(e) \le c(e)$
  - ***Conservation* condition**: For each $v \in V - \{s, t\}$: $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$
- The value of a flow f is $v(f) = \sum_{e \text{ out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



Review: Certificate of Optimality

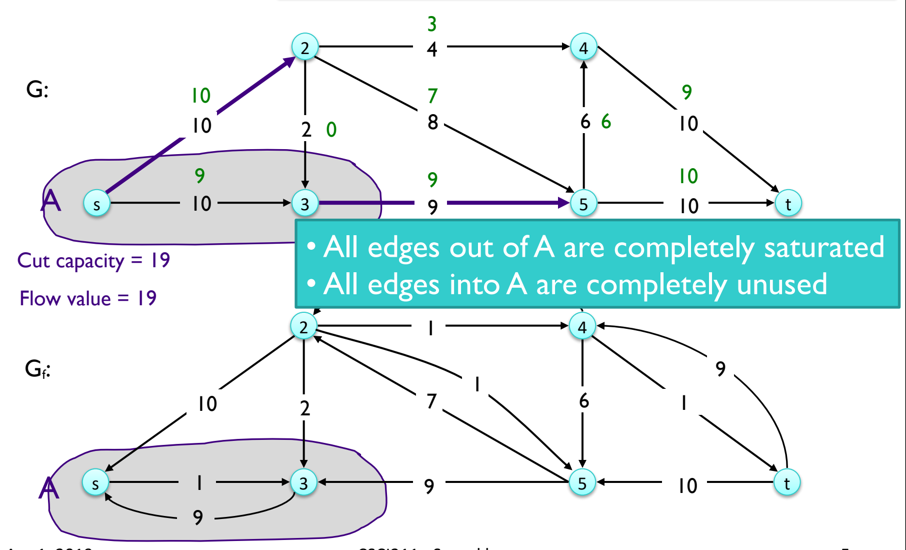
- Corollary. Let *f* be any flow, and let (A, B) be any cut. If v(f) = cap(A, B), then *f* is a max flow and (A, B) is a min cut.

Value of flow = 28
Cut capacity = 28 $\Rightarrow$
Flow value $\le$ 28



Review: Ford-

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



Cut capacity = 19

Flow value = 19

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

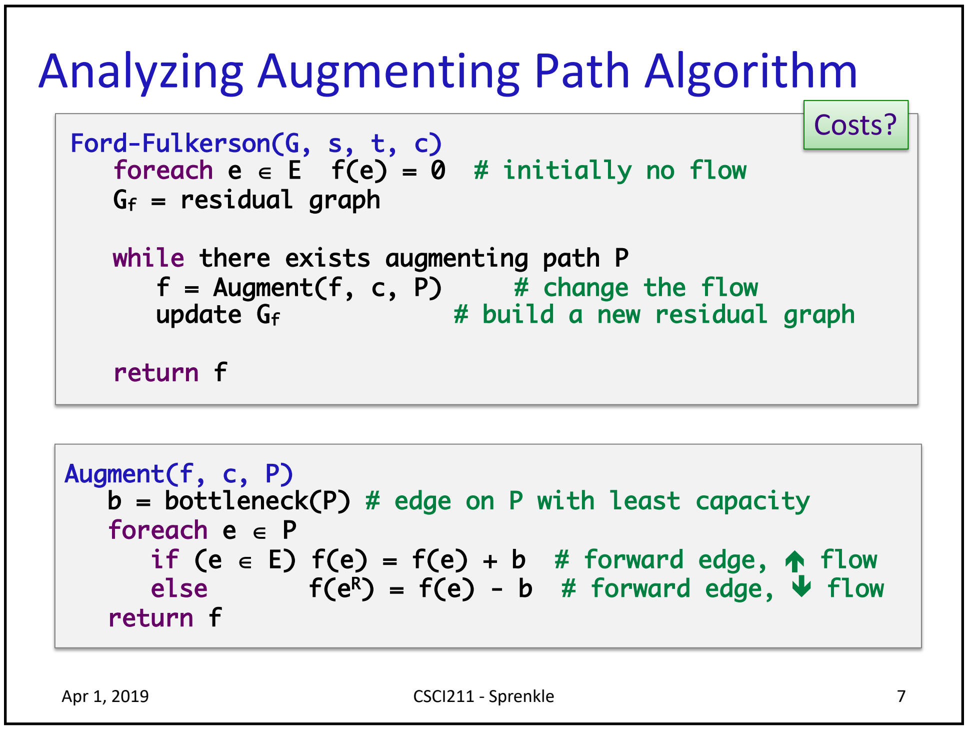
Max-Flow Min-Cut Theorem

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

Costs?

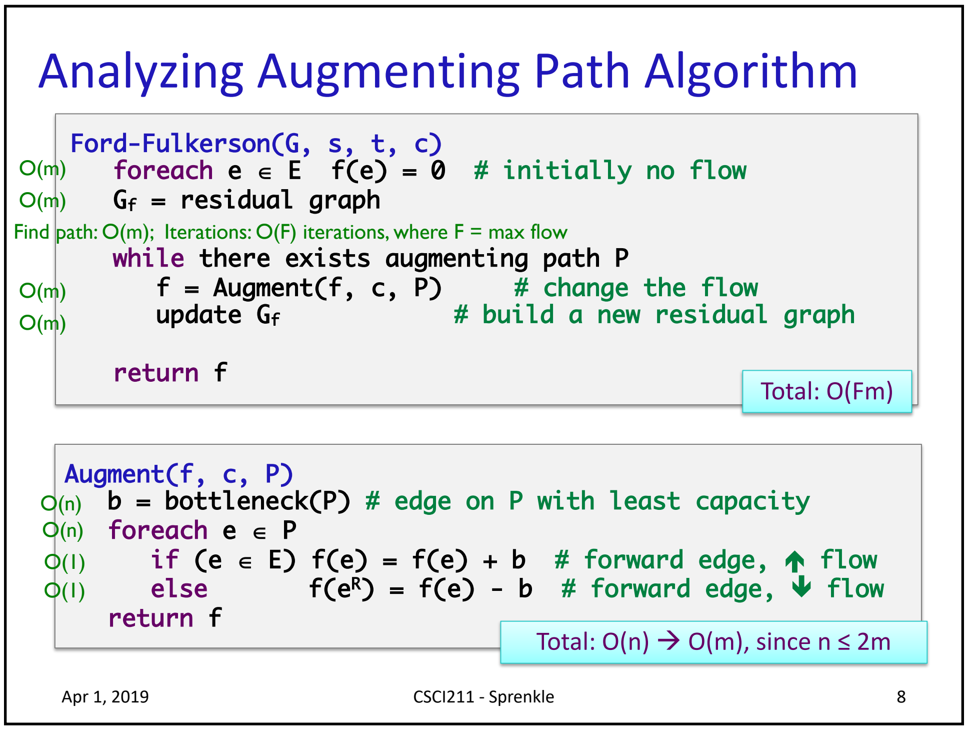
```
Ford-Fulkerson(G, s, t, c)
   foreach e ∈ E  f(e) = 0  # initially no flow
   $G_f$ = residual graph

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

   return f
```

```
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
```

Apr 1, 2019 CSCI211 - Sprenkle 7



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

Apr 1, 2019 CSCI211 - Sprenkle 8

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 $v(f^*) \leq nF$ 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.

Power of Max Flow Problem

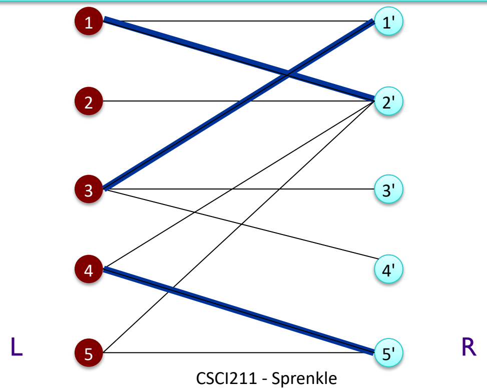
Some problems with non-trivial combinatorial searches can be formulated as **max flow** or **min cut** in a directed graph

BIPARTITE MATCHING

Bipartite Matching

- Input: undirected, bipartite graph G = $(\overbrace{L \cup R}^{V}, E)$
  - Edges: one end in L, one end in R
- Matching M $\subseteq$ E such that each node appears in at most 1 edge in M.

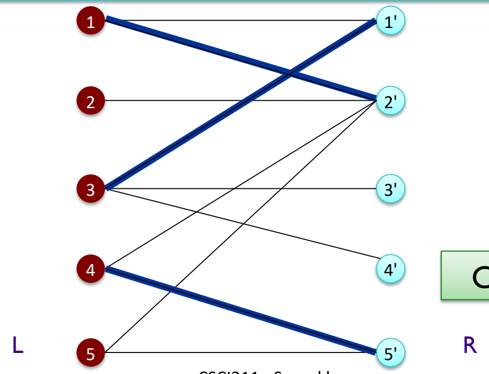
Problem: find matching of largest possible size



Bipartite Matching

- Input: undirected, bipartite graph G = $(\overbrace{L \cup R}^{V}, E)$
  - Edges: one end in L, one end in R
- Matching M $\subseteq$ E such that each node appears in at most 1 edge in M.

Problem: find matching of largest possible size



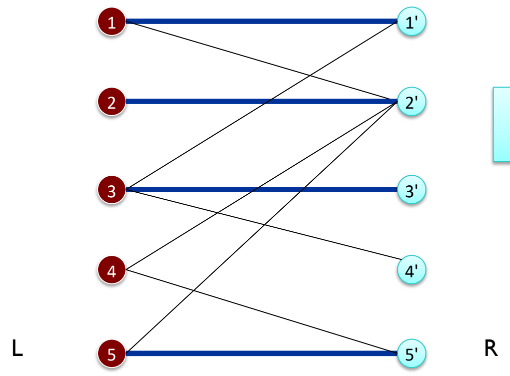
matching

1-2', 3-1', 4-5'

Can we do better?

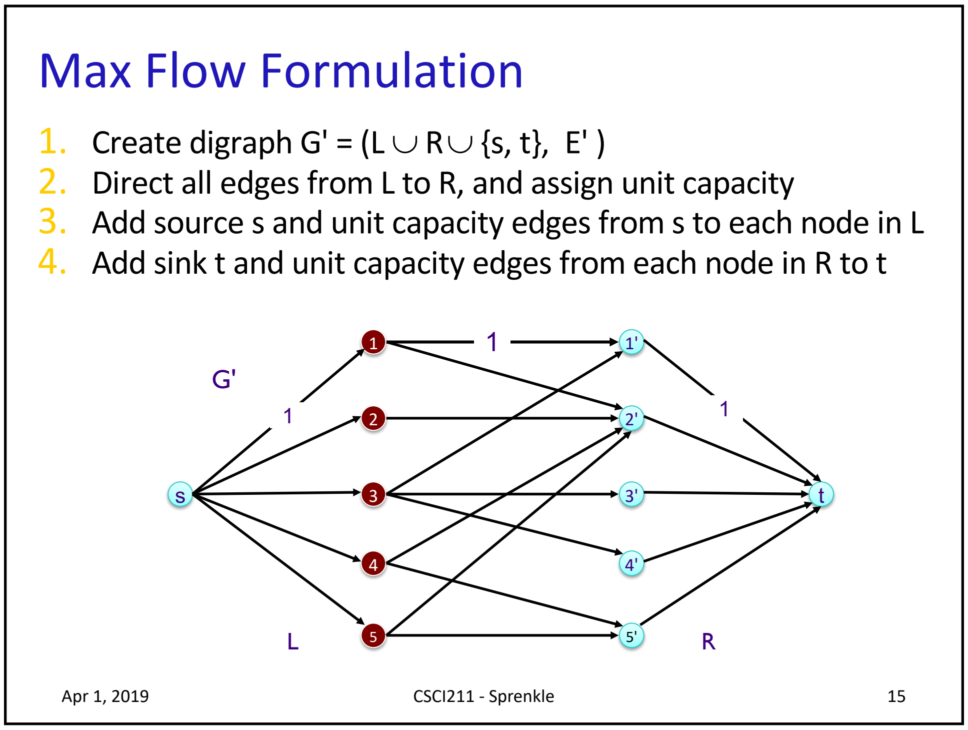
Bipartite Matching

- Input: undirected, bipartite graph G = $(\overbrace{L \cup R}^{V}, E)$
  - Edges: one end in L, one end in R
- Matching M $\subseteq$ E such that each node appears in at most 1 edge in M.



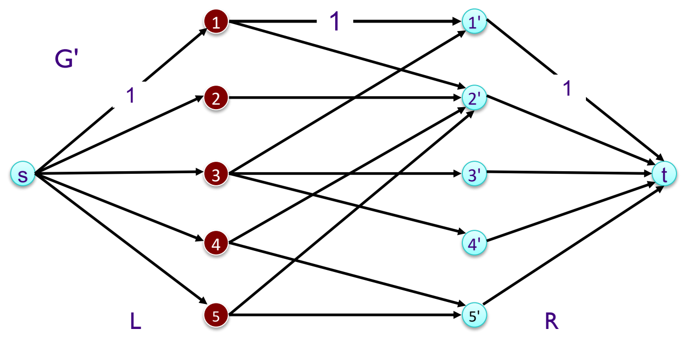
max matching

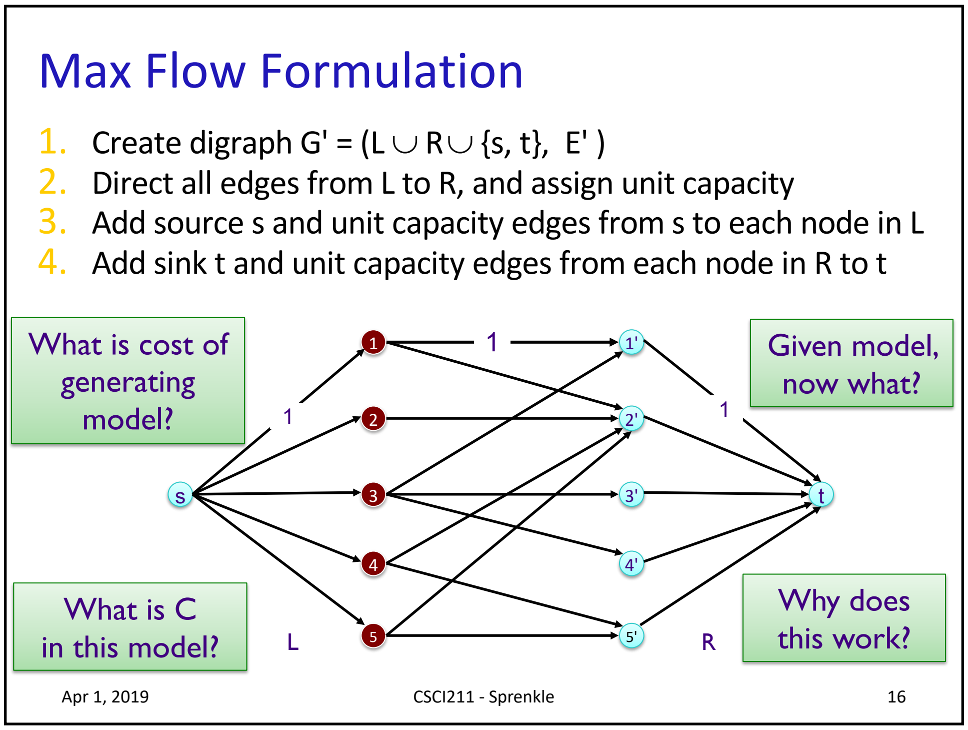
1-1', 2-2', 3-3', 5-5'



Max Flow Formulation

1. Create digraph G' = (L ∪ R∪ {s, t}, E')
2. Direct all edges from L to R, and assign unit capacity
3. Add source s and unit capacity edges from s to each node in L
4. Add sink t and unit capacity edges from each node in R to t



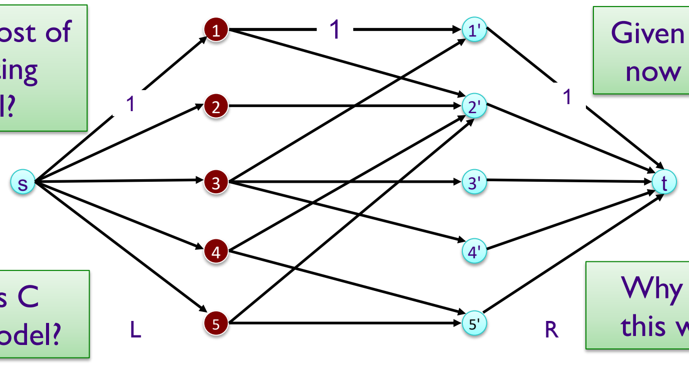


Max Flow Formulation

1. Create digraph G' = (L ∪ R∪ {s, t}, E')
2. Direct all edges from L to R, and assign unit capacity
3. Add source s and unit capacity edges from s to each node in L
4. Add sink t and unit capacity edges from each node in R to t

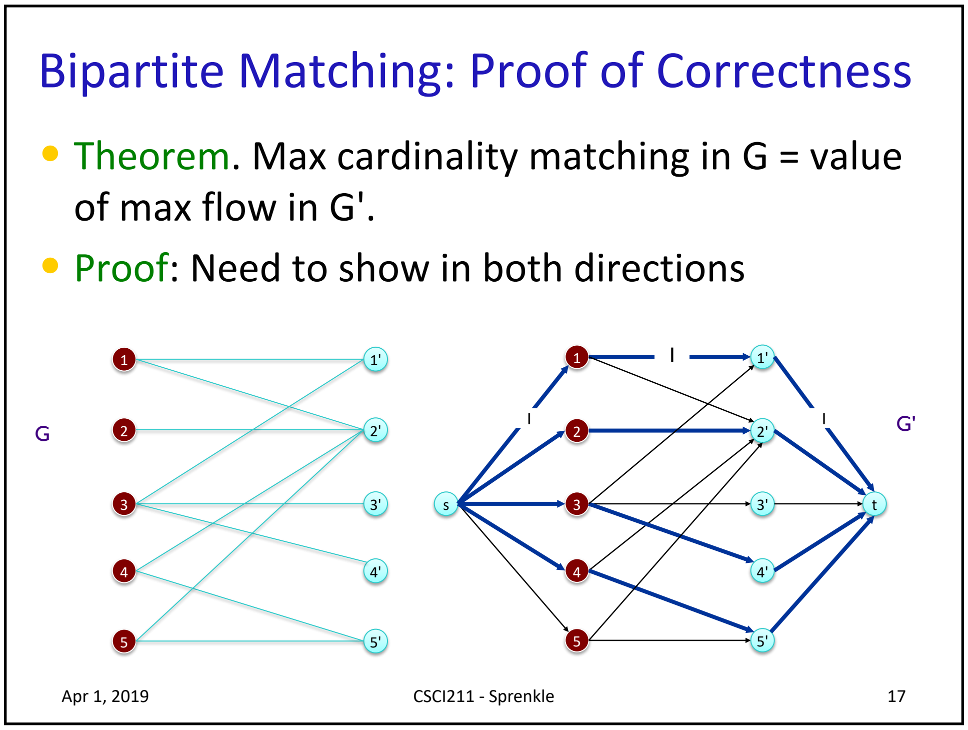
What is cost of generating model?

Given model, now what?



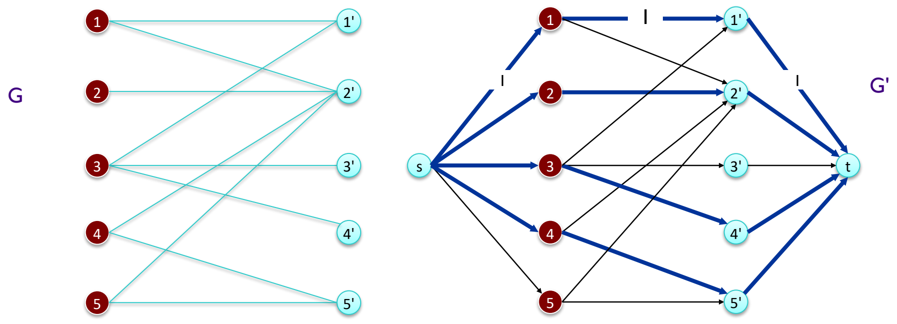
What is C in this model?

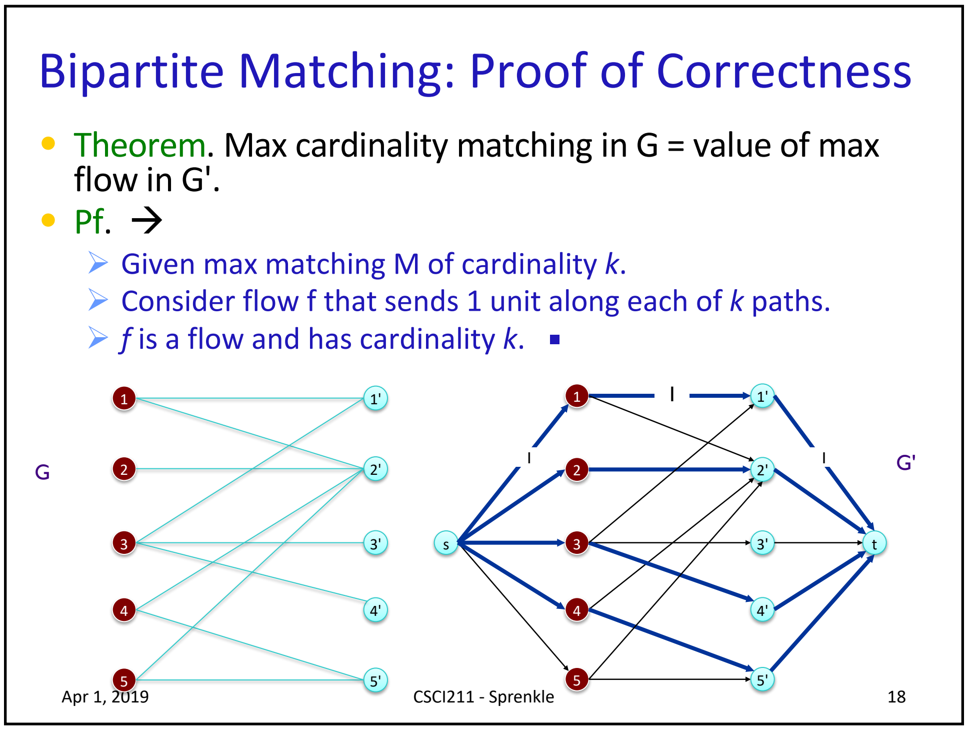
Why does this work?



Bipartite Matching: Proof of Correctness

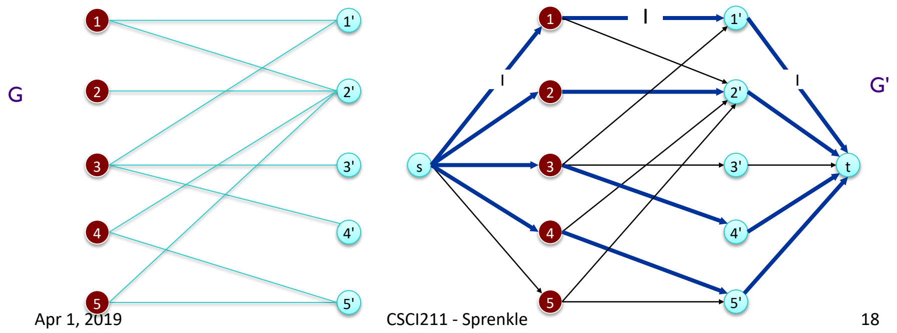
- Theorem. Max cardinality matching in G = value of max flow in G'.
- Proof: Need to show in both directions





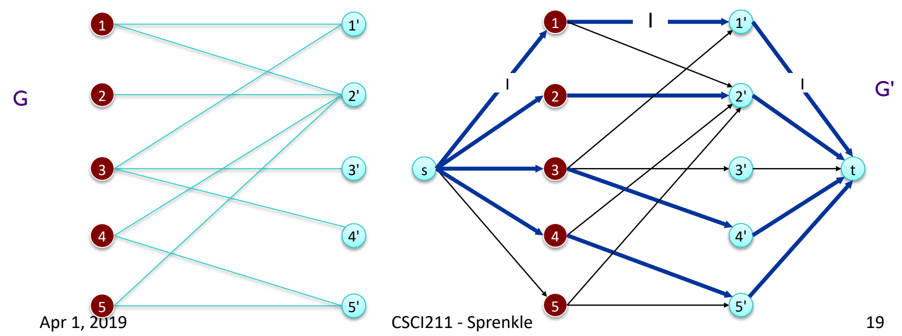
Bipartite Matching: Proof of Correctness

- Theorem. Max cardinality matching in G = value of max flow in G'.
- Pf. →
  - Given max matching M of cardinality *k*.
  - Consider flow f that sends 1 unit along each of *k* paths.
  - *f* is a flow and has cardinality *k*. ▪



Bipartite Matching: Proof of Correctness

- Theorem. Max cardinality matching in G = value of max flow in G'.
- Pf. ←
  - Let f be a max flow in G' of value k.
  - Integrality theorem ⇒ k is integral and can assume f is 0-1.
  - Consider M = set of edges from L to R with f(e) = 1.
    - each node in L and R participates in at most one edge in M
    - |M| = k: consider cut (L ∪ s, R ∪ t) ▪



Network Flow Solutions

1. Model problem as a flow network
   - Describe what nodes, edges, and capacity represent
   - Describe what flow represents and how that maps to your solution
   - Run Ford-Fulkerson algorithm
2. Prove that the solution found is correct/feasible/optimal
3. Prove that you find all solutions
4. Analyze running time
   - Creating model
   - FF algorithm

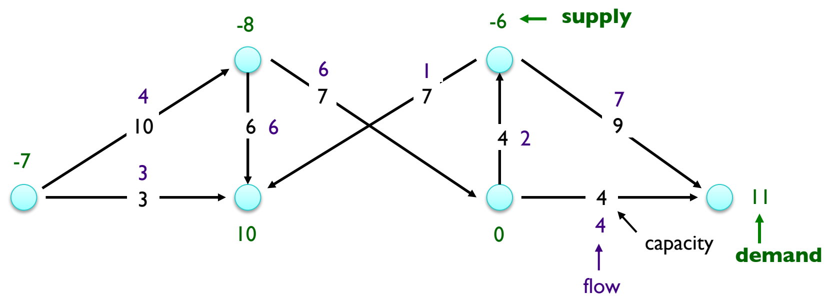
Section 7.7

EXTENSIONS TO MAX FLOW

Circulation with Demands

- Directed graph G = (V, E)
- Edge capacities c(e), e ∈ E
- Node supply and demands d(v), v ∈ V
    - d(v) > 0 → demand
    - d(v) < 0 → supply
    - d(v) = 0 → transshipment

Example Graph: Circulation with Demands



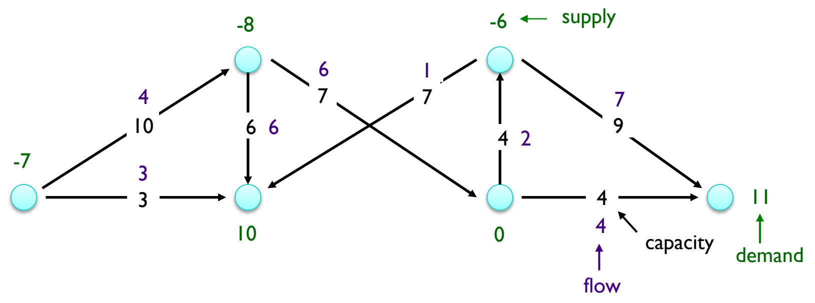
- d(v) > 0 → demand
- d(v) < 0 → supply
- d(v) = 0 → transshipment

Circulation with Demands

- Circulation with demands
  - Directed graph G = (V, E)
  - Edge capacities c(e), e ∈ E
  - Node supply and demands d(v), v ∈ V
- Def. A ***circulation*** is a function that satisfies:
  - For each e ∈ E: $0 \le f(e) \le c(e)$ (capacity)
  - For each v ∈ V: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

Circulation problem:
given (V, E, c, d), does a circulation exist?
(Can we satisfy demand with supply?)

Example Graph: Circulation with Demands



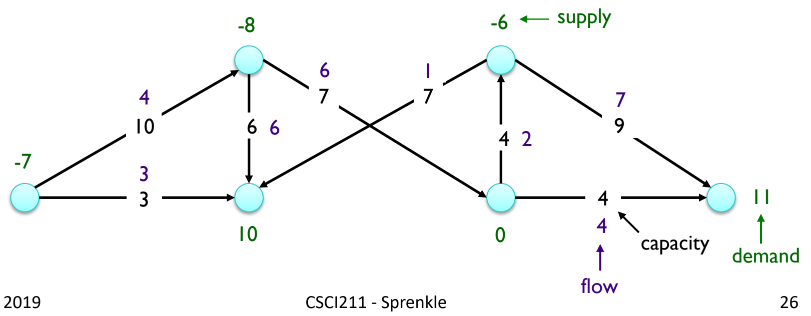
Circulation with Demands

- Necessary condition:

  sum of supplies = sum of demands

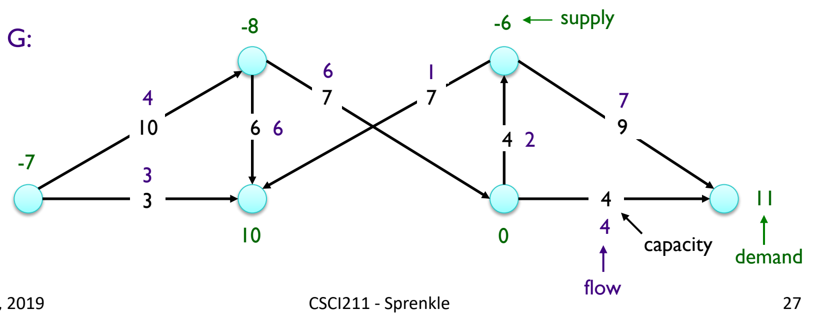
$$\sum_{v:d(v)>0} d(v) = \sum_{v:d(v)<0} -d(v) =: D$$

Sum of supplies? Demands?



Circulation with Demands: Towards Max Flow Formulation

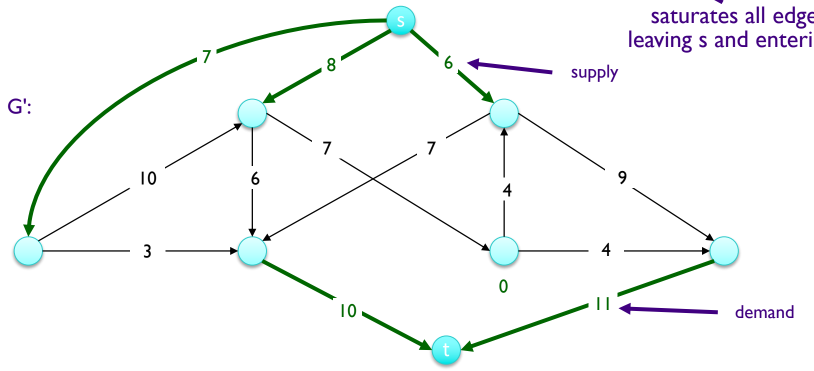
Ideas about how we can formulate this as a max flow problem?

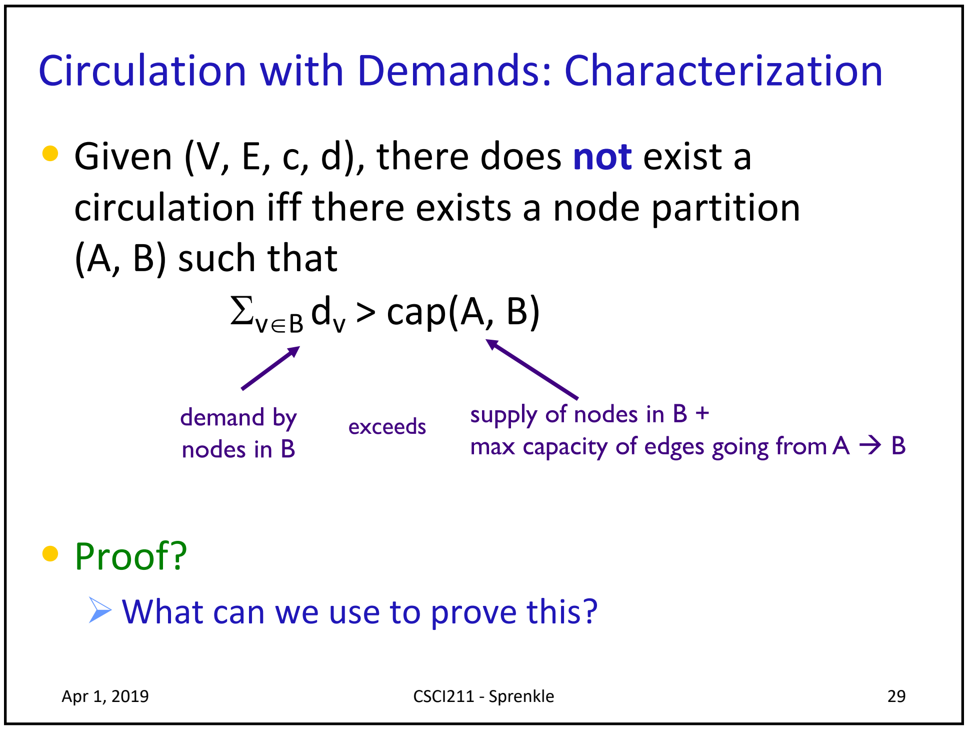


Circulation with Demands: Max Flow Formulation

- Add new source s and sink t
- For each v with d(v) < 0, add edge (s, v) with capacity -d(v)
- For each v with d(v) > 0, add edge (v, t) with capacity d(v)
- **Claim: G has *circulation* iff G' has max flow of value D**

saturates all edges leaving s and entering t





Circulation with Demands: Characterization

- Given (V, E, c, d), there does **not** exist a circulation iff there exists a node partition (A, B) such that

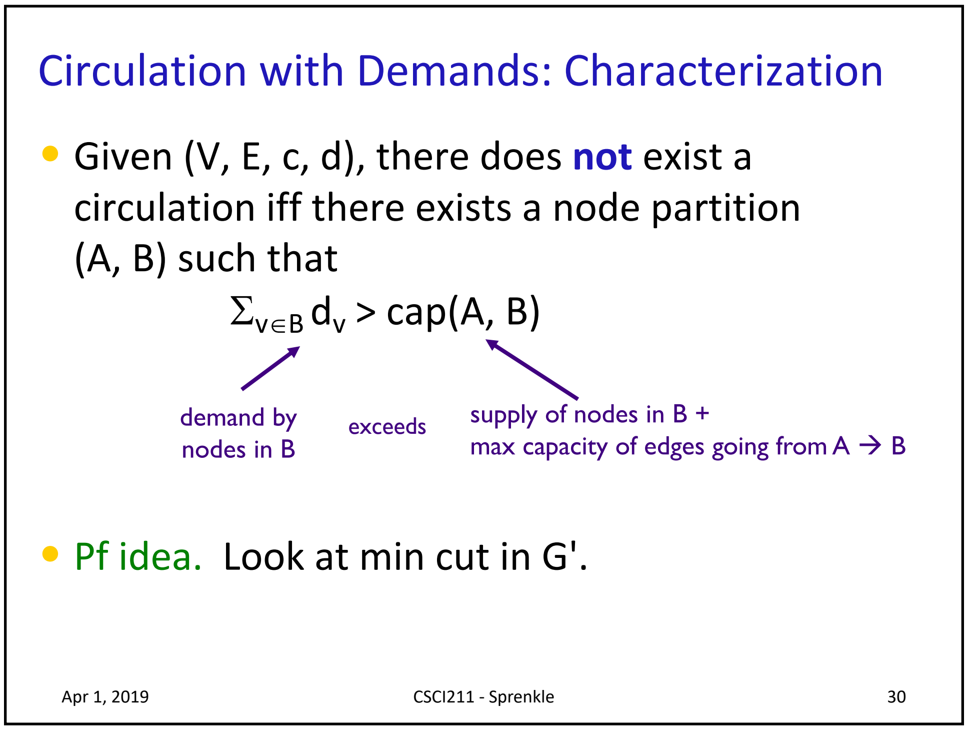
$$\Sigma_{v \in B}\, d_v > \text{cap}(A, B)$$

demand by nodes in B

exceeds

supply of nodes in B + max capacity of edges going from A $\rightarrow$ B

- Proof?
  - What can we use to prove this?



Circulation with Demands: Characterization

- Given (V, E, c, d), there does **not** exist a circulation iff there exists a node partition (A, B) such that

$$\Sigma_{v \in B}\, d_v > \text{cap}(A, B)$$

demand by nodes in B

exceeds

supply of nodes in B + max capacity of edges going from A $\rightarrow$ B

- Pf idea. Look at min cut in G'.

ANOTHER EXTENSION: LOWER BOUNDS

Circulation with Demands and Lower Bounds

- Feasible circulation
  - Directed graph G = (V, E)
  - Edge capacities c(e) and lower bounds ℓ (e), e ∈ E
  - Node supply and demands d(v), v ∈ V

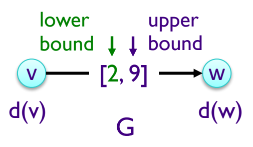
Force flow to use certain edges

- Def. A *circulation* is a function that satisfies:
  - For each e ∈ E: $0 \le \ell(e) \le f(e) \le c(e)$ (capacity)
  - For each v ∈ V: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

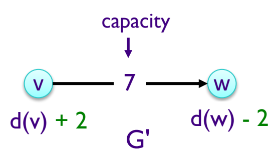
Circulation problem with lower bounds.
Given (V, E, ℓ, c, d), does a circulation exist?

Circulation with Demands and Lower Bounds

- Model lower bounds with demands
  - Send $\ell(e)$ units of flow along edge e
  - Update demands of both endpoints



How we'll represent lower bounds



Supply and demand decrease

Proof in book

7.8 SURVEY DESIGN

Survey Design

- Design survey asking consumers about products
- Can only survey a consumer about a product if they own it
  - Consumer can own multiple products
- Ask consumer i between c_i and c_i' questions
- Ask between p_j and p_j' consumers about product j

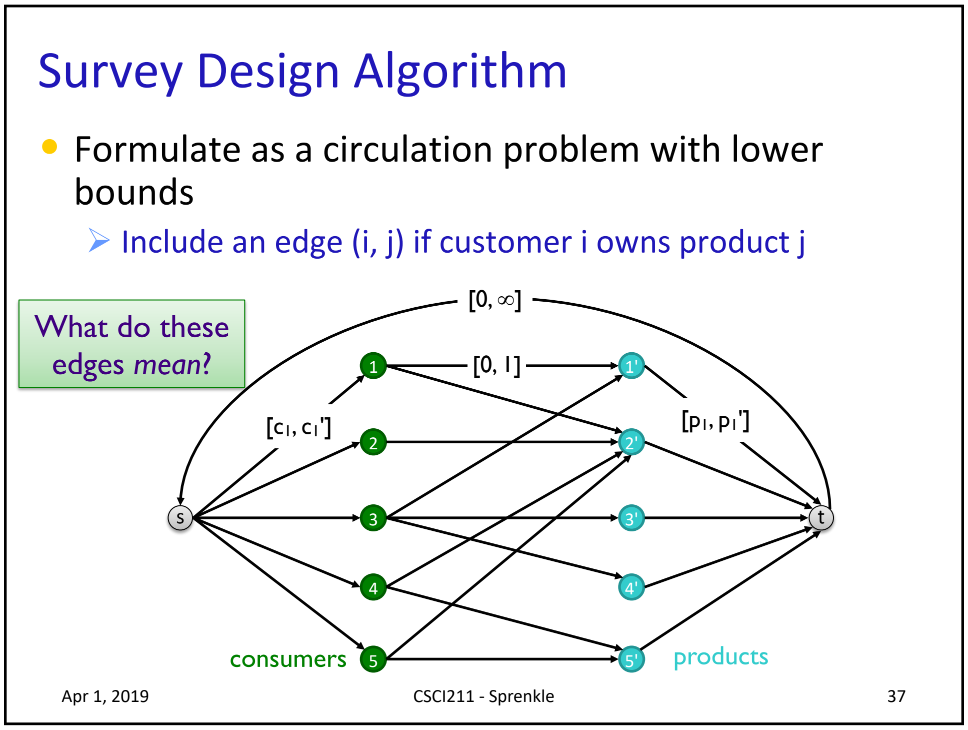
Goal: Design a survey that meets these specs, if possible.

How can we model this problem?

Model: Bipartite Graph

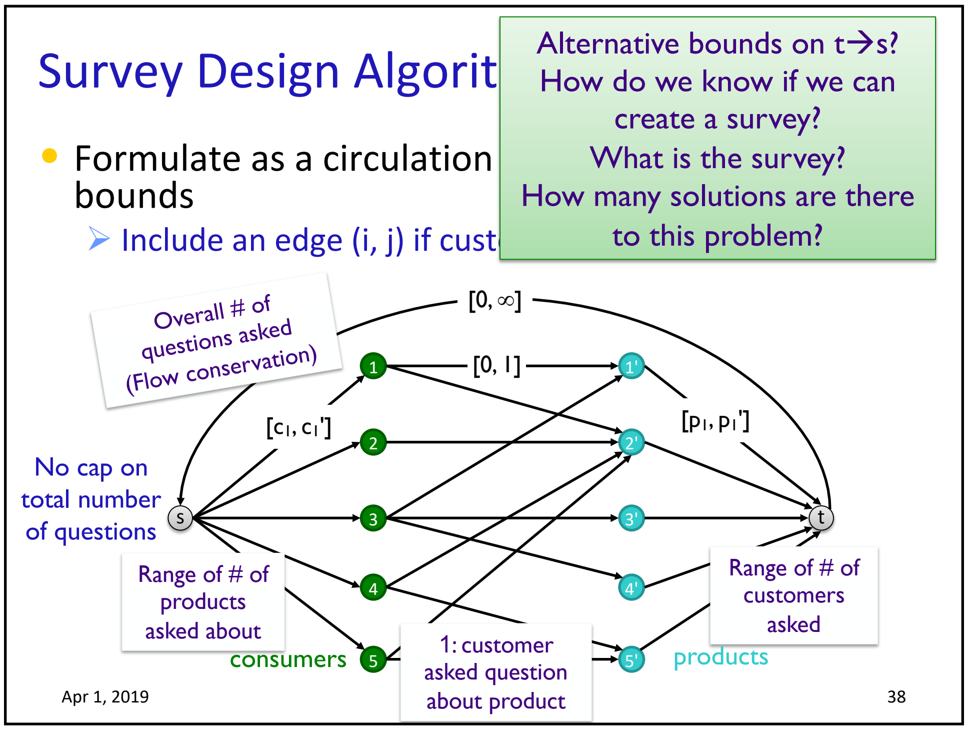
- Nodes: customers and products
- Edge between customer and product means customer owns product
- For each customer, range of # of products asked about
- For each product, range of # of customers asked about it

What does the flow represent?



Survey Design Algorithm

- Formulate as a circulation problem with lower bounds
  - Include an edge (i, j) if customer i owns product j



Survey Design Algorit

- Formulate as a circulation bounds
  - Include an edge (i, j) if cust

Alternative bounds on t→s?
How do we know if we can create a survey?
What is the survey?
How many solutions are there to this problem?

[0, ∞]
[0, 1]
[c1, c1']
[p1, p1']
Overall # of questions asked (Flow conservation)
No cap on total number of questions
Range of # of products asked about
1: customer asked question about product
Range of # of customers asked
s
t
1
2
3
4
5
1'
2'
3'
4'
5'
consumers
products

Survey Solution

- If a feasible, integer flow solution, can create the survey
- Customer *i* will be surveyed about product *j* iff the edge (*i,j*) carries a unit of flow

Survey Solution - Analysis

- How do we know that the solution found is correct/feasible/optimal?
- How do we know that we found all solutions?
- Analyze run time
  - Creating model
  - FF algorithm

Looking Ahead

- Problem Set 9 – due Friday