CSC200: Lecture 19

- Today:
  - Continue pricing algorithm in matching markets
  - Detecting constricted sets and computing perfect matchings
  - Start two sided matching problems Material not in text
  - The stable marriage problem
- Weds and next Monday:
  - Wrap up stable marriage problems
  - Another form of stable matching: kidney exchange
  - Summary of fall term

Announcements

- Final quiz this Friday, December 4 at start of tutorial; scope will be one-sided matching markets and market clearing prices.
- Final lecture of term Monday, December 7

Computing Market Clearing Prices

- Why do market clearing prices always exist?
- We'll construct an *algorithm* that computes market clearing prices [Demange, Gale, Sotomayor, 1986]
- It will be a type of "ascending auction" for multiple items
  - sellers gradually raise their prices (starting from zero)
  - buyers indicate willingness to buy at current prices (pref. seller graph)
  - if any set of items is "overdemanded" (i.e., constricted set), the sellers of those items can raise their prices
    - forces some buyers to consider other items (reduce contention)
  - we'll continue until a perfect matching is found in the PSG
  - we'll add a simple price adjustment step (gives us *minimal* prices)
- It is really nothing more than a multi-item auction (but with different sellers *coordinating* their price adjustments explicitly)

A Simple, Coordinated Multi-item Auction

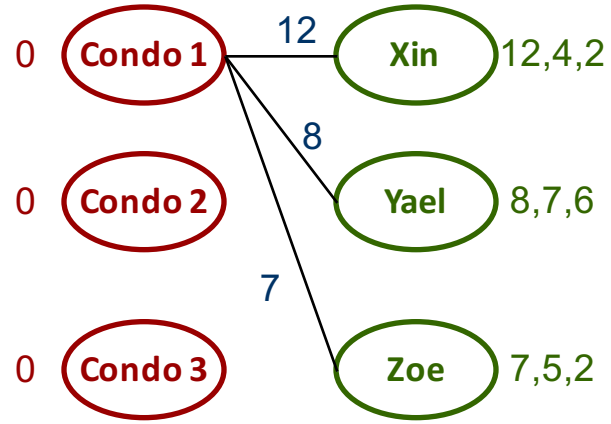
0. Each seller sets initial price of zero
1. Construct the preferred seller graph (PSG) using current prices
   - *buyers valuations may be known, or they bid for all preferred items*
2. Check if there is a perfect matching in this PSG
   a. If so, **stop**: current prices are market clearing (and matching is one reasonable assignment of items to buyers)
   b. If not, continue to next step
3. Find a constricted set of buyers *S*, and neighboring sellers *N(S)*
   - *in fact, we would like to find a minimal constricted set*
4. Raise price of each item in *N(S)*, *and only items in N(S),* by one
5. If price of each item is greater than zero, reduce each price by the amount of the lowest price (i.e., make lowest price zero)

   Aside: I don't think this step is necessary since lowest price never more than 0 (!)
6. Return to Step 1 and repeat

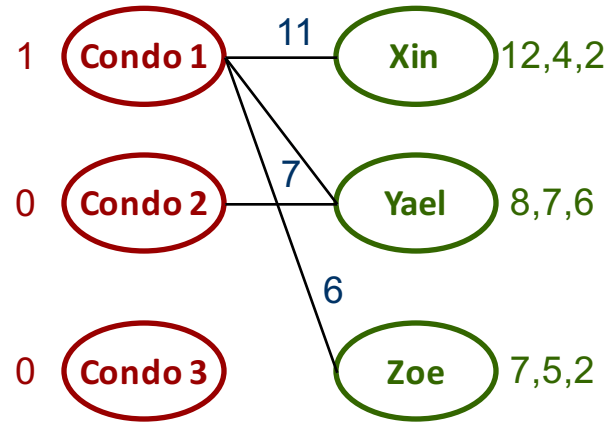
Rough Intuitions

- Steps 1 and 2:
  - at current prices, if perfect matching is found, then we obviously have market clearing prices
- Step 3:
  - if no perfect matching, must have a constricted set: need to fix it
  - in economic terms, the set *N(S)* is *overdemanded*
- Step 4
  - if *S* is a constricted set, there is too much competition for *N(S)* at the current prices for the item in *N(S)*
  - by raising prices of each item in *N(S)*, we may make items outside of *N(S)* more attractive to some buyers in *S* (eventually this must happen!)
- Step 5
  - Possibly use a price adjustment ensures lowest price is always zero
  - price adjustment does not affect the PSG at all (if *y* preferred by *x* before, still preferred when all prices decreased uniformly)

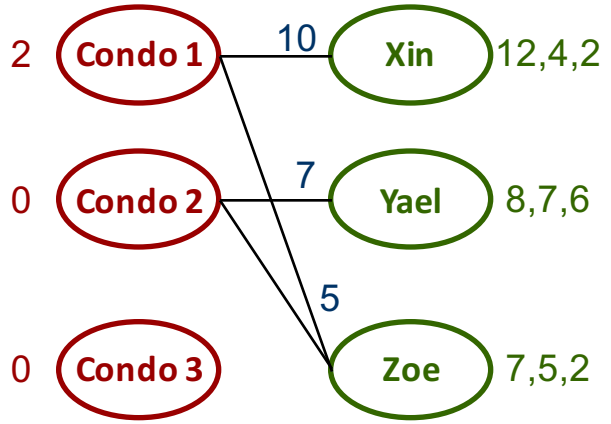
Illustration



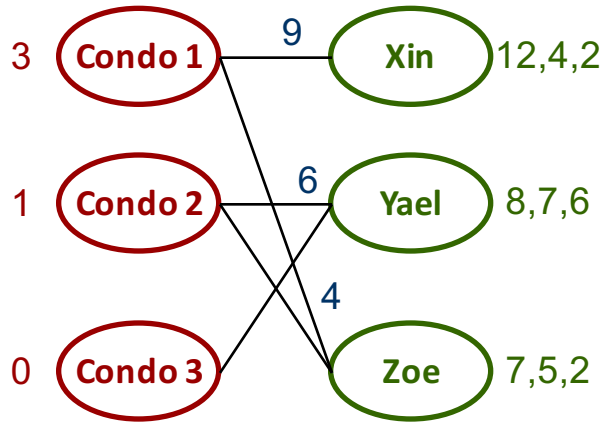
Increase Price of C1



Increase Price of C1 *(or C1, C2)*



Increase Price of C1, C2



Market Clears

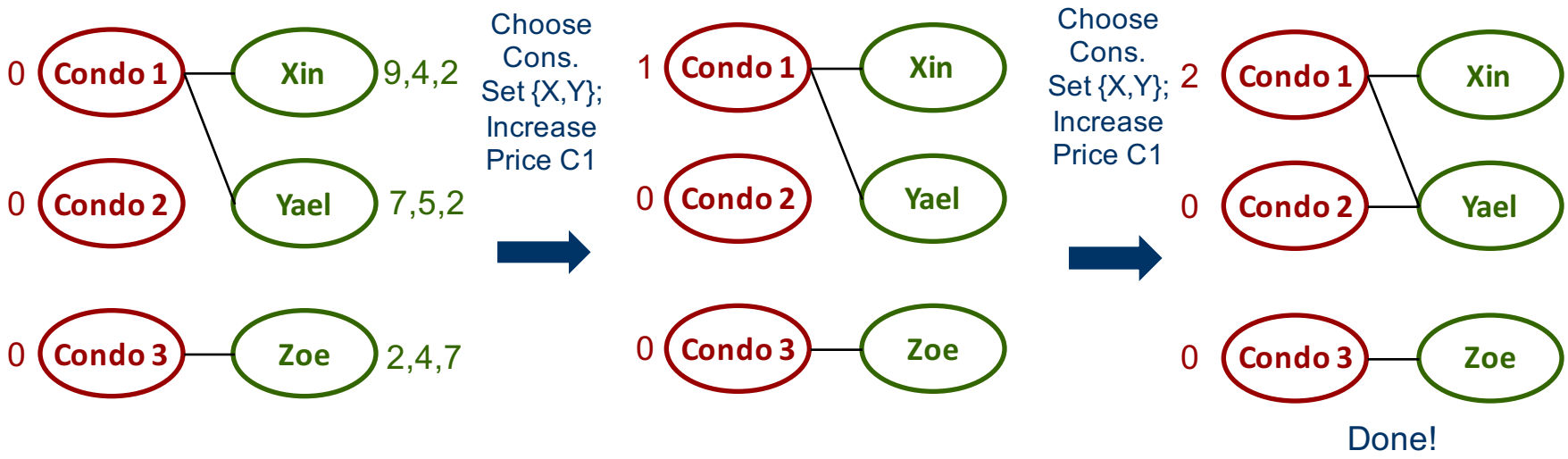
The Auction Must Converge (1)

- A simple "potential" argument can be used to show auction converges
- Given current set of prices, define
  - *potential of buyer:* maximum utility she can derive (item that maximizes valuation minus price); this is value of any edge in PSG
  - *potential of a seller:* utility obtained if matched, i.e., current price
  - *potential of auction:* sum of all buyer, seller potentials
    - note: this is *not* yet max social welfare, because it assumes each buyer gets most preferred item at current prices (even if it is overdemanded)
- All buyers start with potential equal to value of best item
- All sellers start with potential equal to zero
- So total auction potential at first stage is P_0 (sum of best item values)
  - again note: this is *not* maximum social welfare

The Auction Must Converge (2)

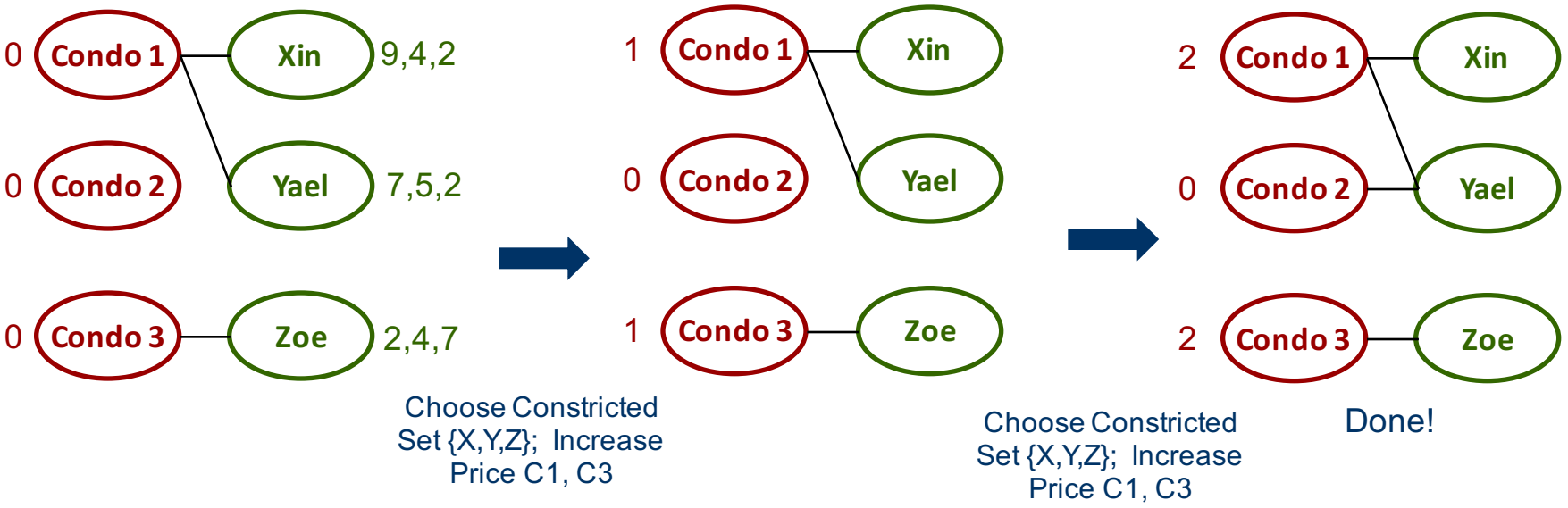
- What happens to the total potential at each step of the auction?
- When prices of items in *N(S)* increase for some constricted set *S*, the total potential goes down by at least one. Why?
  - There are *n* buyers in *S*; and *N(S)* contains all of their most preferred items: so their maximum utility decreases by 1 each (total reduction in potential of *n*)
  - There are *m* items in *N(S)*, and each seller has their utility increase by 1 (total increase in potential of *m*)
  - But since *m* is less than *n* (since *S* is constricted), if we don't stop auction with a perfect matching, total potential decreases by at least 1
  - Nothing else changes potential (including resetting lowest price to zero, since buyer/seller changes cancel out)
  - So the auction can go on for at most P_0 steps

An Example: Different Choices of *S*



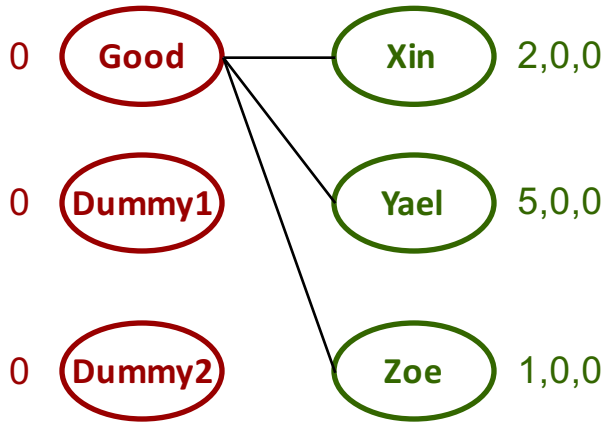
- When we choose *minimal* constricted sets, the prices we end up at are the *minimal market clearing prices* (Demange, Gale, Sotomayor)
  - any other market clearing prices must have prices at least as high for each item (we won't prove this!)
  - it turns out that these are VCG prices for a special case where buyers only want one item, aka *unit-demand auctions*): see Ch.15 (advanced)
- Other choices of constricted sets lead to different prices (next slide)

An Example: Different Choices of *S*

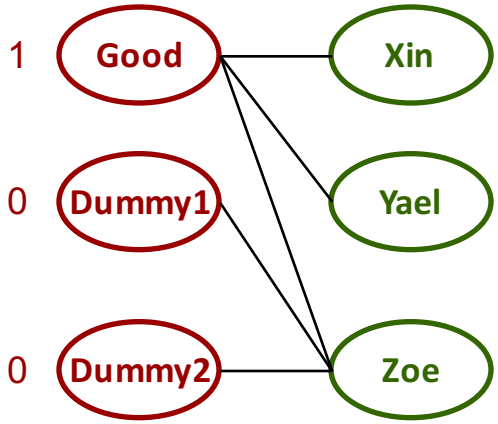


Relation to English/Vickrey Auction

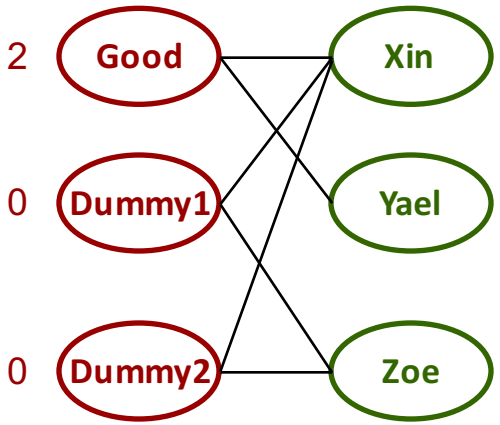
- Easy to see how English (ascending) auction for a single good works as an example:
  - one item (seller), *n* bidders
  - create *n-1* dummy goods (value zero to all bidders)
  - bidder *b* connected only to item in PSG until price increases to *b*'s value
  - final price is value of second highest value bidder, and perfect matching awards item to highest value bidder (i.e., VCG price)



Round 1



Round 2

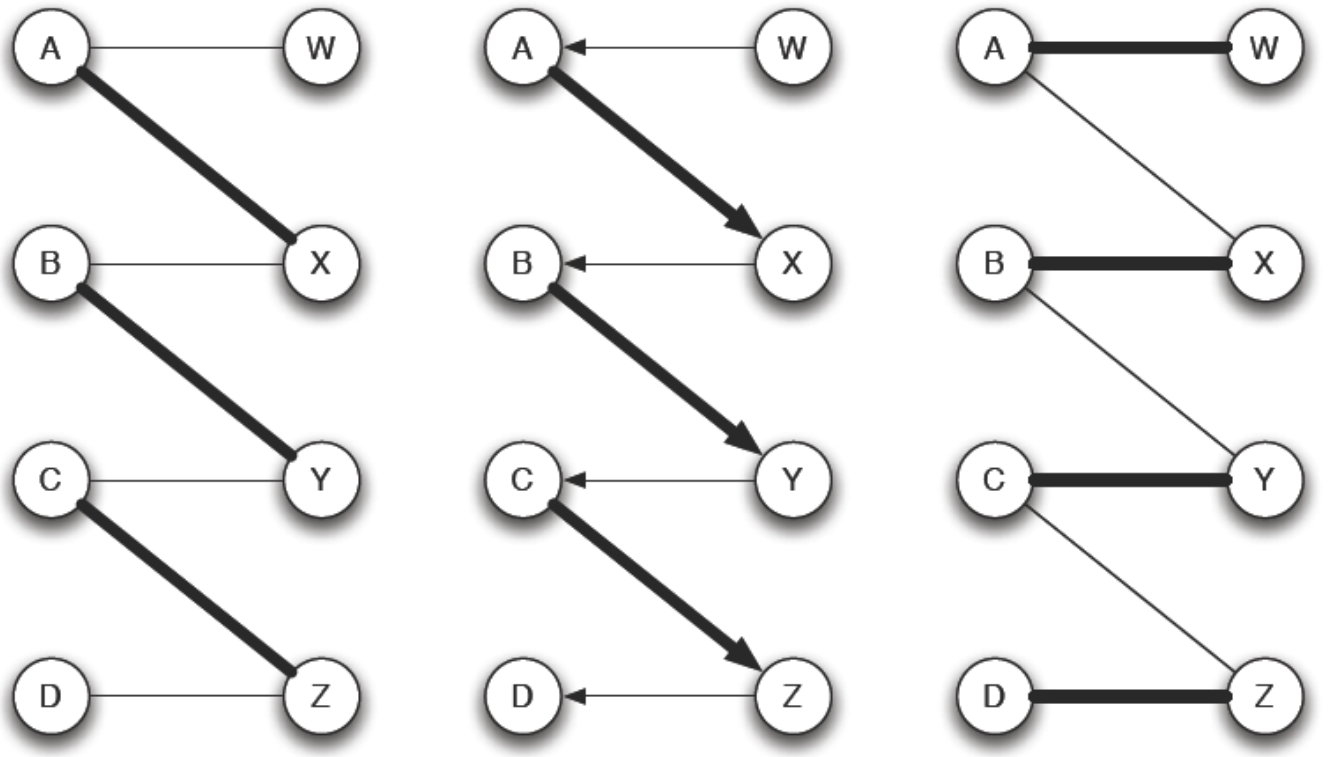


Round 3: Done!

Finding Perfect Matchings/Constricted Sets

- Only missing piece is how to find perfect matching and constricted sets.
- We'll discuss general picture, but call the nodes on the right "agents" and the ones on the left "items" (like in matching markets)
- Very quick review
  - start with the empty matching and proceed in stages
  - At any stage, assume we have a matching *M* that is not perfect
  - Try find an *augmenting path* for *M*: this is a path that starts at an unmatched agent *X*, then connects to an unmatched item *Y* via an *alternating path*: edges out of matching, edge in matching, edge out of matching…
  - If we find one, replace all the matched edges with the unmatched edges: increases matching size by one pair
  - If we don't find it, we've found a constricted set

Augmenting Path: Example



(a) *A matching that is not of maximum size*

(b) *An augmenting path*

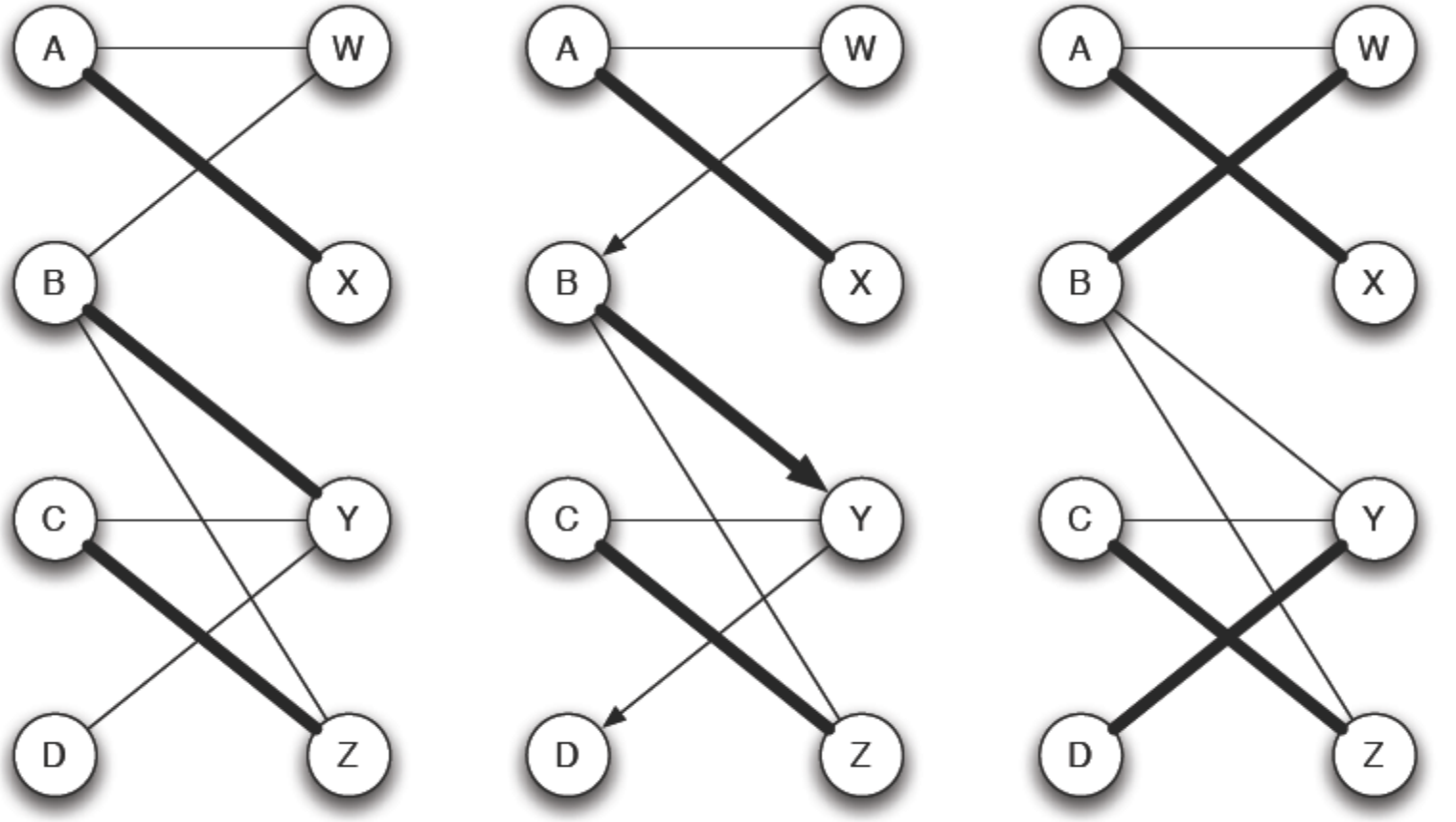
(c) *A larger (perfect) matching*

[E&K, Ch.10, Fig 10.9]

- The alternating path from W to D shows how to "augment" the initial matching: gives a matching that is one pair (item-agent) bigger

Another Example

[E&K, Ch.10, Fig 10.10]



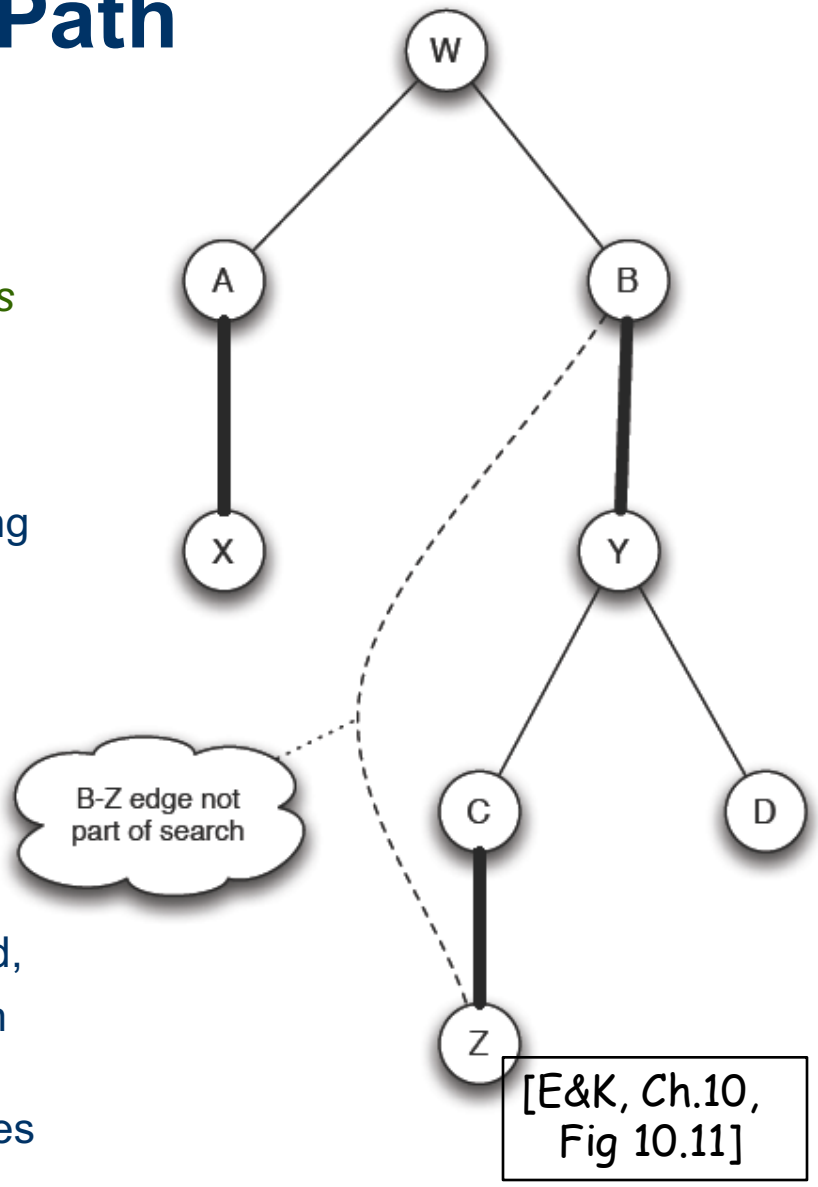
(a) *A matching that is not of maximum size*

(b) *An augmenting path*

(c) *A larger (perfect) matching*

Search for Alternating Path

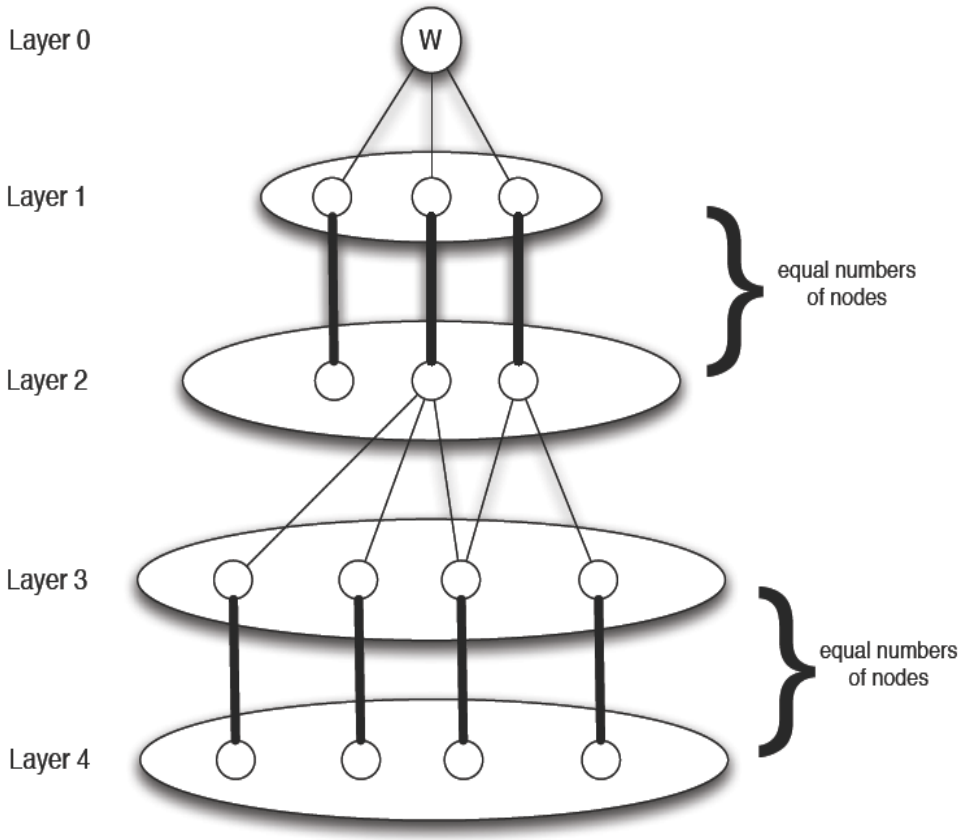
- To find an augmenting path from W
  - Start a breadth-first search from W
  - At first level, can only connect using edges not in matching *(*since W unmatched, that's all of W's edges)*
  - At second level, can only connect using edges in matching
  - At third level, only use edges not in matching
  - And so on…
  - … never including duplicate nodes
- If you reach an unmatched node, you have found an augmenting path
- Convince yourself that:
  - Apart from W (root), only agents ever included in search tree are already matched,
  - Hence only unmatched elements apart from W are items
  - Hence, any unmatched items must be leaves in the search tree



Proof of Matching Theorem

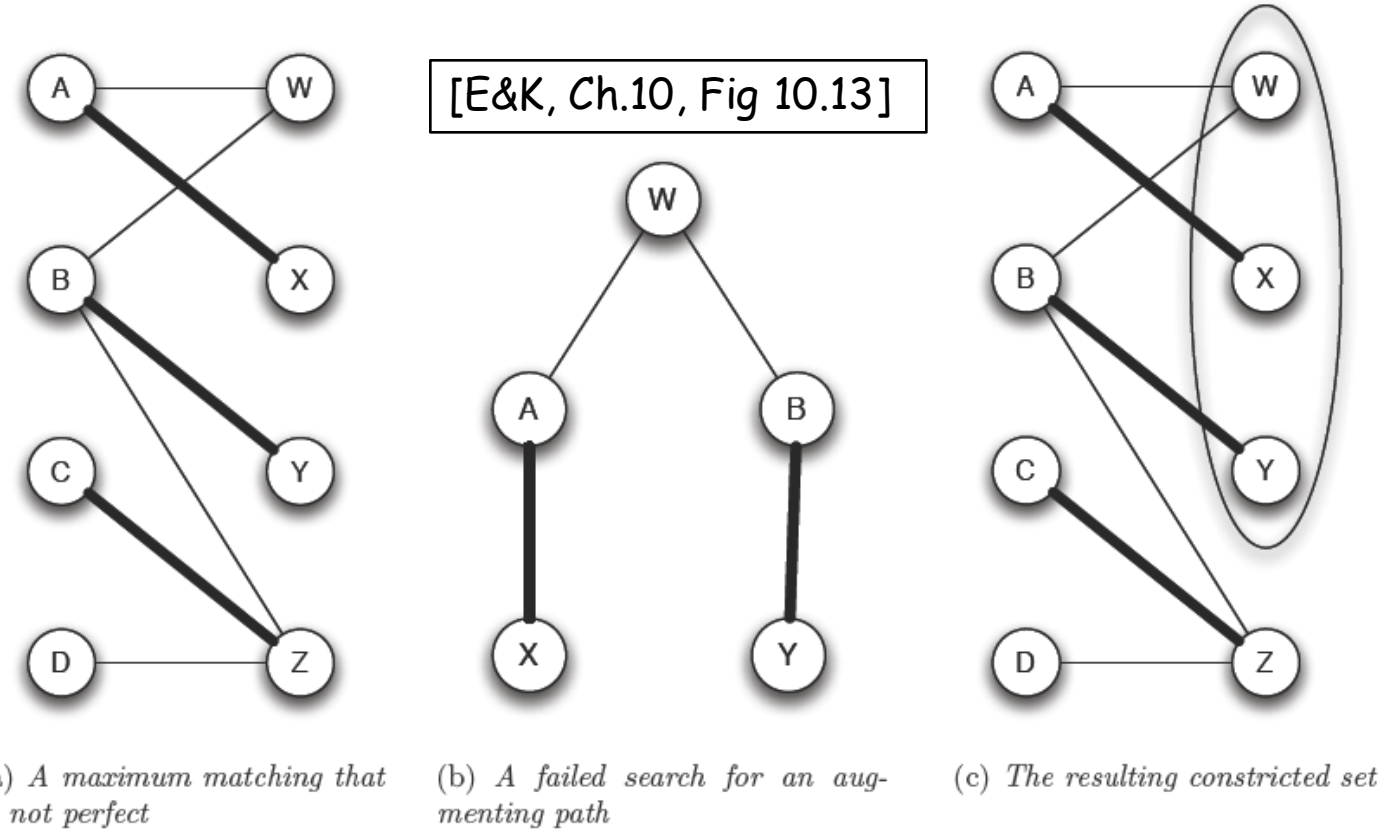
- If search terminates and some leaf is an unmatched item, we have an augmenting path

- If every leaf is a matched agent:
  - each layer of agents (even layers), *except root (W)* has exactly as many agents as items in the preceding layer
  - if an agent appears in the search tree, so do all of the items it is connected to in the bipartite graph (or PSG):
    - its matched neighbor is in the layer above
    - all its unmatched neighbors are in the layer below
- So we have a set of a set of agents that is exactly one bigger than its set of neighboring items: *constricted set!*



[E&K, Ch.10, Fig 10.12]

An Illustration



(a) *A maximum matching that is not perfect*

(b) *A failed search for an augmenting path*

(c) *The resulting constricted set*

- BFS for augmented path terminates: all leaves are matched agents
- So the set of agents in the search tree (W, X, Y) form a constricted set: all of their neighbors (A, B) are in the tree too

Summary

- Put all this together: a simple algorithm for finding a *maximum matching M* (which is *perfect*, if one exists)
  - For each unmatched agent node *X*, try to find an augmenting path from *X* (using depth first search until an unmatched item is reached); add first augmenting path found to *M*
  - Repeat this until no augmenting paths found
  - Search for augmenting path takes *O(m)* time (*m* = #edges)
  - Repeats at most *n/2* times (once an agent is matched, never becomes unmatched) (*n* = #vertices)
  - Whole process takes *O(mn)* time
- If you search for multiple augmenting paths at once, its faster: $O(m\sqrt{n})$
- Aside: The augmenting path algorithm(s) for maximum matching in a bipartite graph is a special case of the Ford Fulkerson maximum flow algorithm(s).

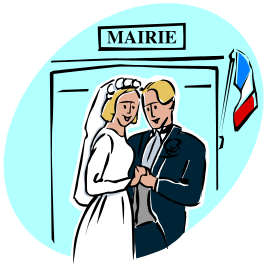
Two-sided Matching Markets

- *Two-sided (single-unit) matching markets:*
  - two sets of agents, *X* and *Y*
    - *X* and *Y* could be the same set in some instances
  - any *x* in *X* has preferences over *Y*, any *y* in *Y* has preferences over *X*
  - goal: match agents in *X* to agents in *Y* to satisfy some objective
  - unlike one-sided, where "condos" didn't care who they were matched to
- Examples:
  - hiring workers/interns (companies *X* and interviewees *Y*)
  - assigning grad students to supervisors (seriously, at Duke!)
  - assign tasks *X* to employees *Y* (tasks prefer better qualified employees)
  - matching men *X* and women *Y* (classic: *stable marriage problem*)
  - assign pairs of students to *shared* dorm rooms
    - basically, matching students with each other (*X*=*Y*)
- Today: we'll start the study of the *stable marriage problem*

Stable Marriage Problem

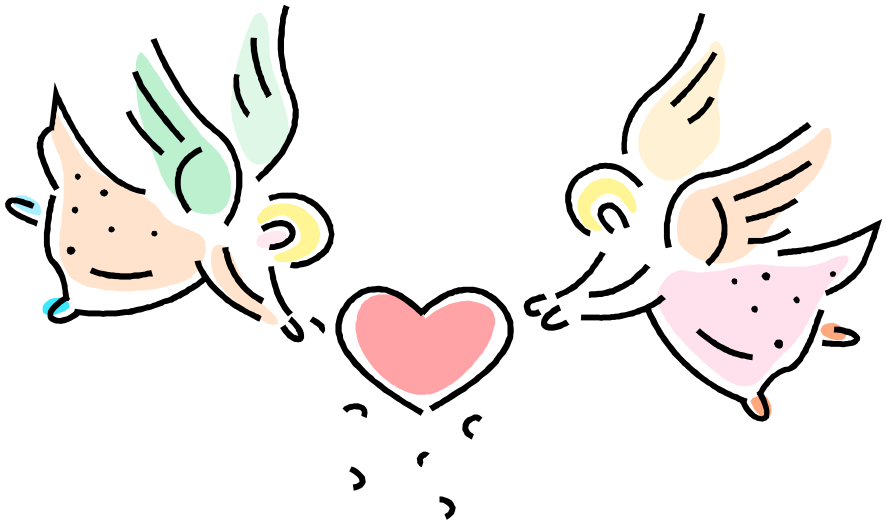
- Classic model proposed by Gale, Shapley (1960)
- Assume a set of *n* men *M*, *n* women *W*
- Each man has *preferences* over the set of women, and each women has *preferences* over the set of men
- Since we don't imagine people always being able to say how much a potential partner is "worth", we'll assume that preferences are represented by a *preference ordering* or *ranking*

- Model (and algorithm) can be applied to any 1:1 matching problem with preferences on each side of matching
  - matching roommates; students and supervisors; jobs with interns, residents with hospitals, etc…



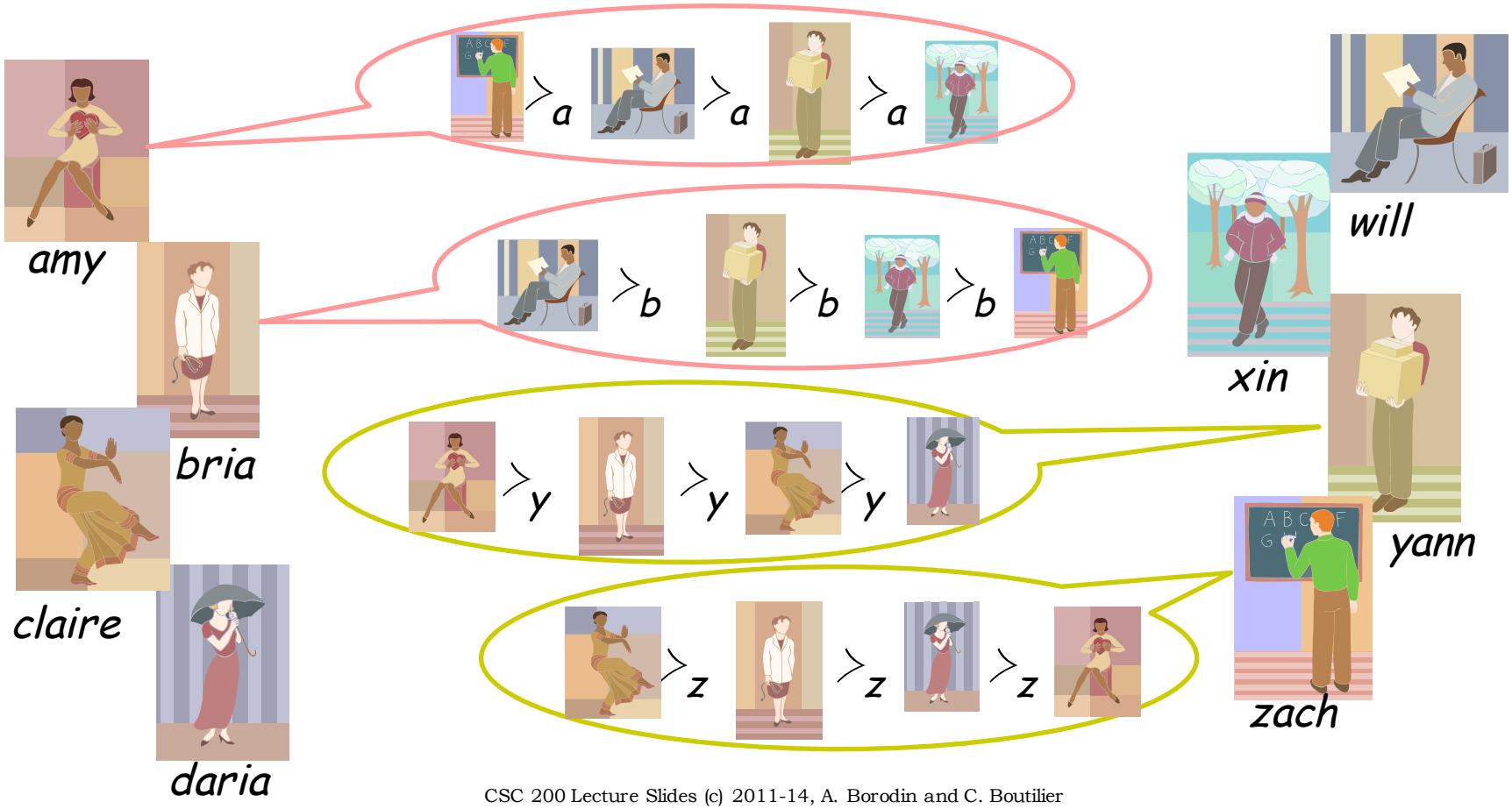
Preference Orderings

- Preference orderings: just a ranked list of potential partners
  - Each man m has preference ordering $\succ_m$ over W
  - Each woman w has preference ordering $\succ_w$ over M



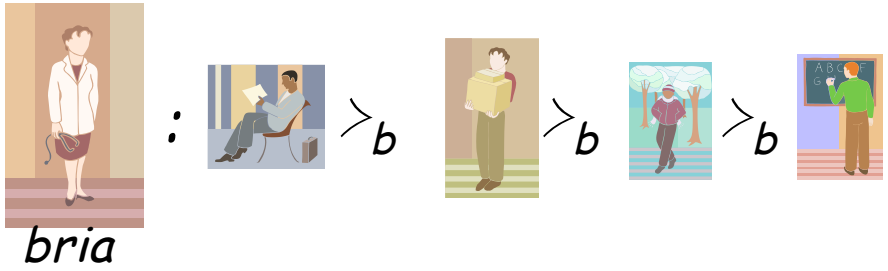
Preference Orderings

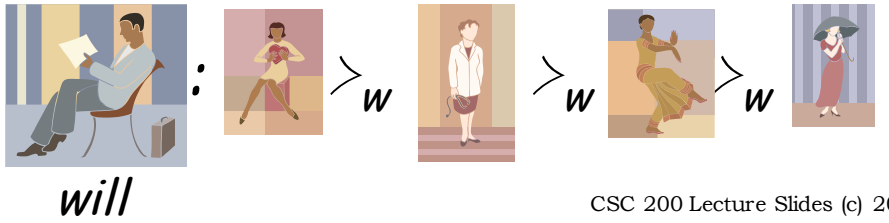
- Preference orderings: just a ranked list of potential partners
  - Each man m has preference ordering $\succ_m$ over W
  - Each woman w has preference ordering $\succ_w$ over M

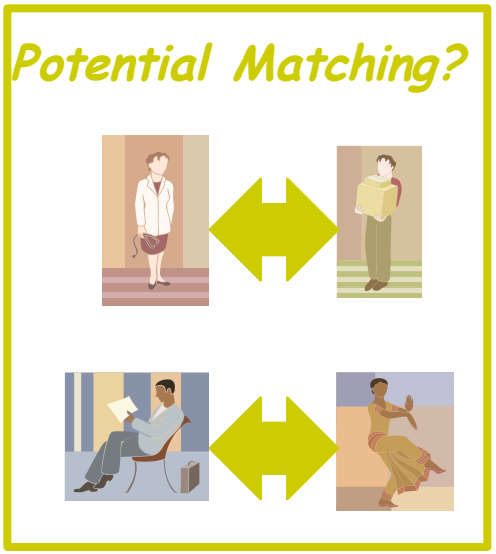


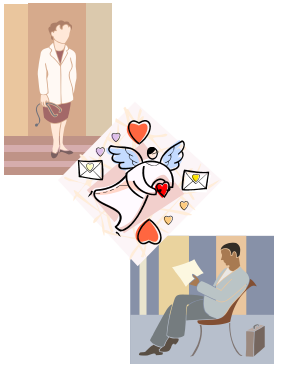
Stable Matchings (Marriage)

- Goal: find a matching of men and women that is desirable
  - maximize social welfare? don't have utilities to measure
  - it should at least be Pareto optimal
  - we'd like matching to be "stable"
- Intuitively, if we find a matching where some man *m* prefers a woman *w* to his "matched partner", and *w* prefers *m* to her "matched partner", they would "run off" with each other
- A *stable matching* should avoid that!









Stable Matching/Marriage

- A *matching (marriage)* is a mapping μ that associates a women with each man and a man with each women
  - denote the partner of man m by $\mu(m) \in W$
  - denote the partner of woman w by $\mu(w) \in M$
  - we require that $\mu(m) = w$ iff $\mu(w) = m$
  - *Just a set of pairs (m,w) where each man, women is in exactly one pair*

- A pair *(m,w) blocks* matching μ if $w \succ_m \mu(m)$ and $m \succ_w \mu(w)$
  - Both m and w would prefer to be with each other than with the partners assigned to them by μ

- Matching is *stable* iff it is unblocked by any pair
  - In other words, while some man m might prefer a *different* partner w, his preferred partner w does *not* prefer m to her partner (and similarly for women)

Some Simple Examples

Man	1st	2nd	3rd
x	a	b	c
y	b	a	c
z	a	b	c

Woman	1st	2nd	3rd
a	y	x	z
b	x	y	z
c	x	y	z

Match 1

a-x
b-y
c-z

It is stable (x,y don't want to move; a, b won't move except to x,y; so c,z stuck)

Match 2

a-y
b-x
c-z

It is stable (a,b don't want to move; x, y won't move except to a,b; so c,z stuck)

Match 3

a-z
b-y
c-x

Unstable (b,x form a blocking pair)

(a,x also form a blocking pair)