Lec # 26, 27 & 28

Augmenting Path Algorithm

Input: Bipartite graph $G = (X \cup Y, E)$.

Steps:

1. Find an arbitrary matching M.
2. Let U denote the set of unsaturated vertices in X.
3. If U is empty, then M is a maximum matching; otherwise, select a vertex x from U.
4. Consider y in $N(x)$.
5. If y is also unsaturated by M, then add the edge xy to M to obtain a larger matching M'. Return to Step (2) and recompute U. Otherwise, go to Step (6).
6. If y is saturated by M, then find a maximal M-alternating path from x using xy as the first edge.

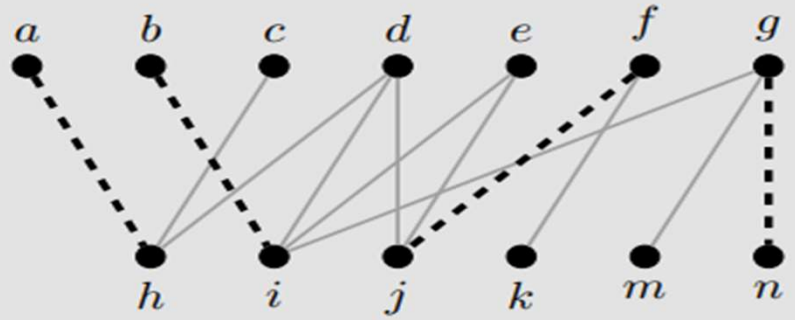
(a) If this path is M-augmenting, then switch edges along that path to obtain a larger matching M'; that is, remove from M the matched edges along the path and add the unmatched edges to create M'. Return to Step (2) and recompute U.

(b) If the path is not M-augmenting, return to Step (4), choosing a new vertex from $N(x)$.

7. Stop repeating Steps (2)–(4) when all vertices from U have been considered.

Output: Maximum matching for G.

Example 5.5 Apply the Augmenting Path Algorithm to the bipartite graph G_6 below, where $X = \{a, b, c, d, e, f, g\}$ and $Y = \{h, i, j, k, m, n\}$, with an initial matching shown as dashed lines.

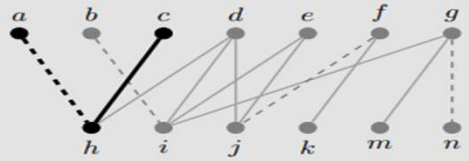


G_6

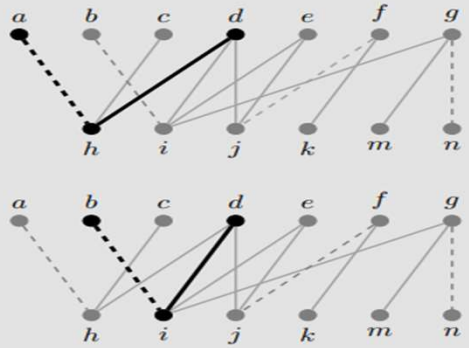
Solution: During each step shown below, the path under consideration will be in bold, with the matching shown as dashed lines throughout.

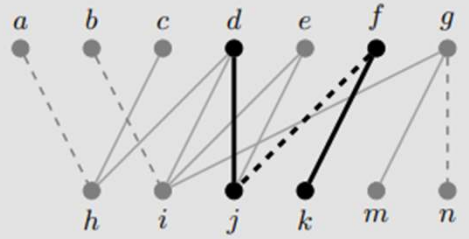
Step 1: The unsaturated vertices from X are $U = \{c, d, e\}$.

Step 2: Choose c. The only neighbor of c is h, which is saturated by M. Form an M-alternating path starting with the edge ch. This produces the path $c\,h\,a$, as shown on the next page, which is not augmenting.

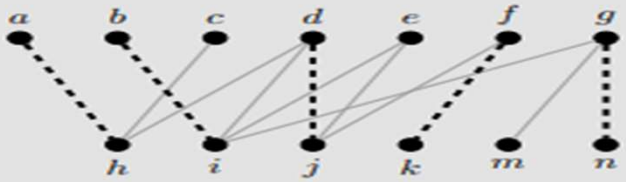


Step 3: Choose a new vertex from U, say d. Then $N(d) = \{h, i, j\}$. Below are the alternating paths originating from d.



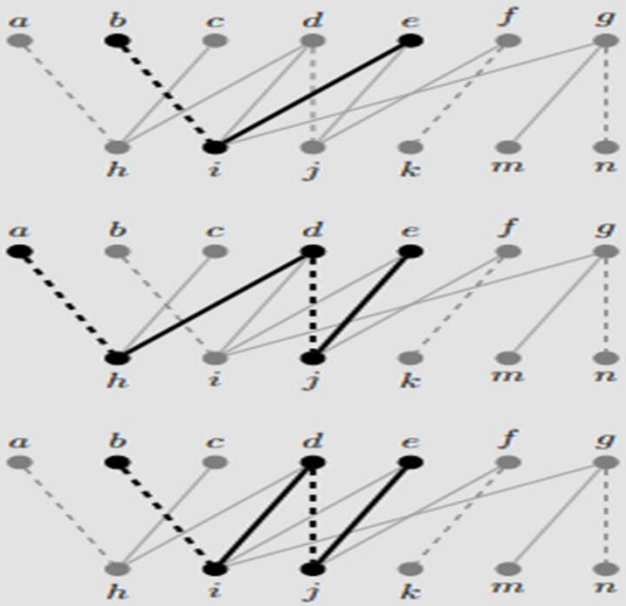


Note that the last path $(d\,j\,f\,k)$ is M-augmenting. Form a new matching M' by removing edge fj from M and adding edges dj and fk, as shown in the following graph.



Step 4: Recalculate $U = \{c, e\}$. We must still check c since it is possible for the change in matching to modify possible alternating paths from a previously reviewed vertex; however, the path obtained is $c\,h\,a$, the same as from Step 2.

Step 5: Check the paths from e. The alternating paths are shown below.



None of these paths are augmenting. Thus no M'-augmenting paths exist in G and so M' must be maximum by Berge's Theorem.

Output: The maximum matching $M' = \{ah, bi, dj, fk, gn\}$ from Step 3.

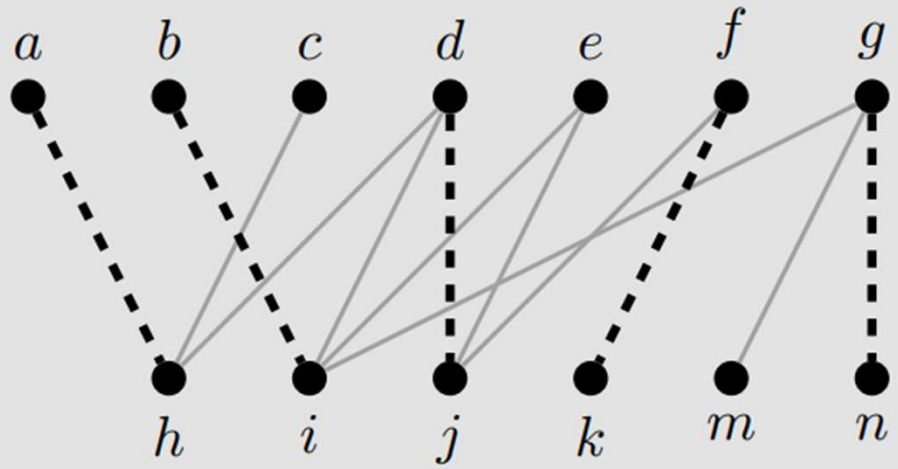
Definition 5.10 A ***vertex cover*** Q for a graph G is a subset of vertices so that every edge of G has at least one endpoint in Q.

Theorem 5.11 (König-Egerváry Theorem) For a bipartite graph G, the size of a maximum matching of G equals the size of a minimum vertex cover for G.

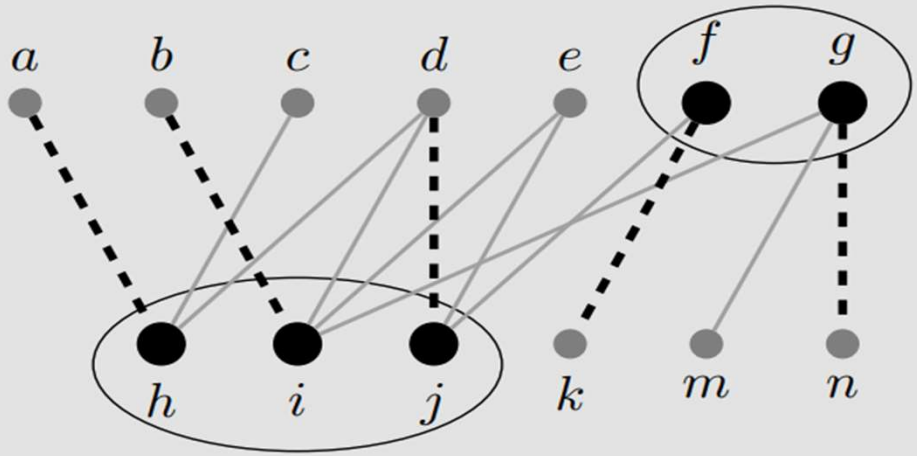
Vertex Cover Method

1. Let $G = (X \cup Y, E)$ be a bipartite graph.
2. Apply the Augmenting Path Algorithm and mark the vertices considered throughout its final implementation.
3. Define a vertex cover Q as the unmarked vertices from X and the marked vertices from Y.
4. Q is a minimum vertex cover for G.

Example 5.6 Apply the Vertex Cover Method to the output graph from Example 5.5.



Solution: Recording the vertices considered throughout the last step of the Augmenting Path Algorithm, the marked vertices from X are a, b, c, d, and e, and the marked vertices from Y are h, i, and j. This produces the vertex cover $Q = \{f, g, h, i, j\}$ of size 5, shown below. Recall that the maximum matching contained 5 edges.



Matching in General Graphs

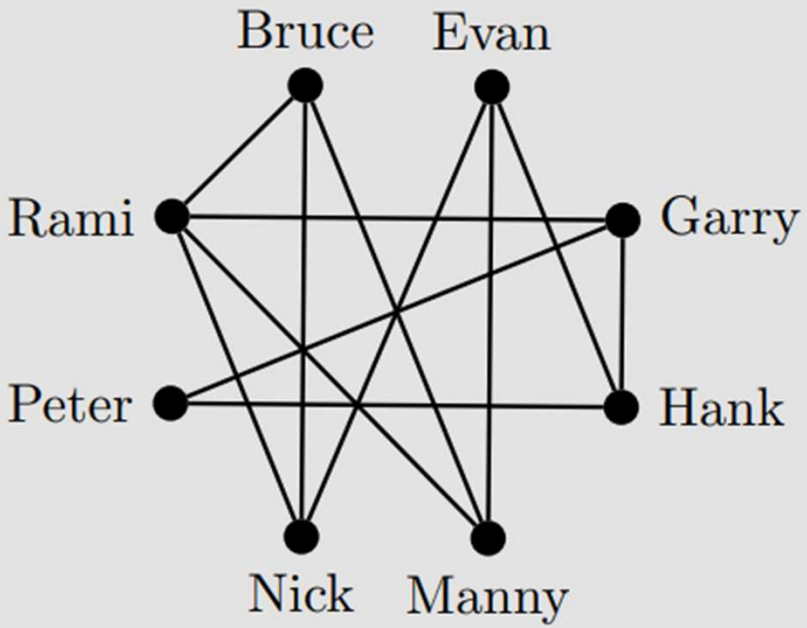
Theorem 5.7 (Berge's Theorem) A matching M of a graph G is maximum if and only if G does not contain any M-augmenting paths.

Bruce, Evan, Garry, Hank, Manny, Nick, Peter, and Rami decide to go on a week-long canoe trip in Guatemala. They must divide themselves into pairs, one pair for each of four canoes, where everyone is only willing to share a canoe with a few of the other travelers.

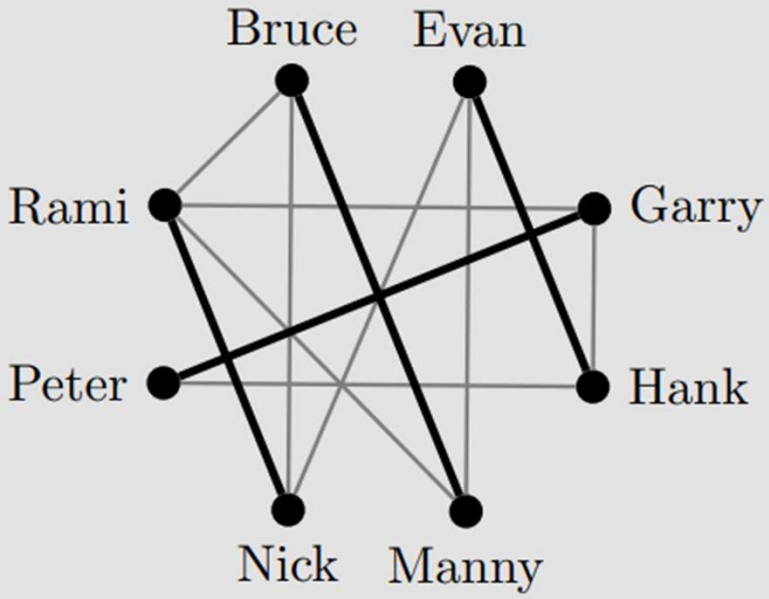
Example 5.9 The group of eight men from above have listed who they are willing to share a canoe with. This information is shown in the following table, where a Y indicates a possible pair. Note that these relationships are symmetric, so if Bruce will share a canoe with Manny, then Manny is also willing to share a canoe with Bruce. Model this information as a graph. Find a perfect matching or explain why no such matching exists.

	Bruce	Evan	Garry	Hank	Manny	Nick	Peter	Rami
Bruce	·	·	·	·	Y	Y	·	Y
Evan	·	·	·	Y	Y	Y	·	·
Garry	·	·	·	Y	·	·	Y	Y
Hank	·	Y	Y	·	·	·	Y	·
Manny	Y	Y	·	·	·	·	·	Y
Nick	Y	Y	·	·	·	·	·	Y
Peter	·	·	Y	Y	·	·	·	·
Rami	Y	·	Y	·	Y	Y	·	·

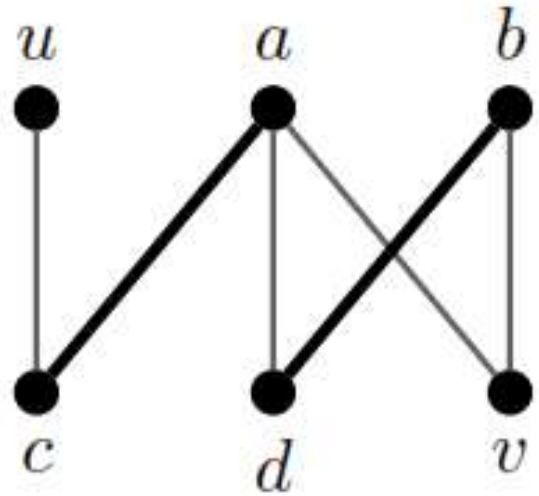
Solution: The graph is shown below where an edge represents a potential pairing into a canoe.



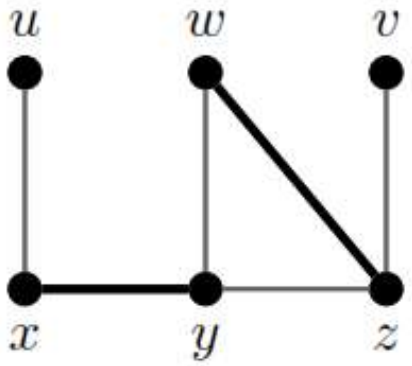
Note that Peter can only be paired with either Garry or Hank. If we choose to pair Peter and Garry, then Hank must be paired with Evan, leaving Nick and Manny to each be paired with one of Rami and Bruce. One possible matching is shown below. Since all people have been paired, we have a perfect matching.



Corollary 5.16 Every cubic graph without any bridges has a perfect matching.



G_8



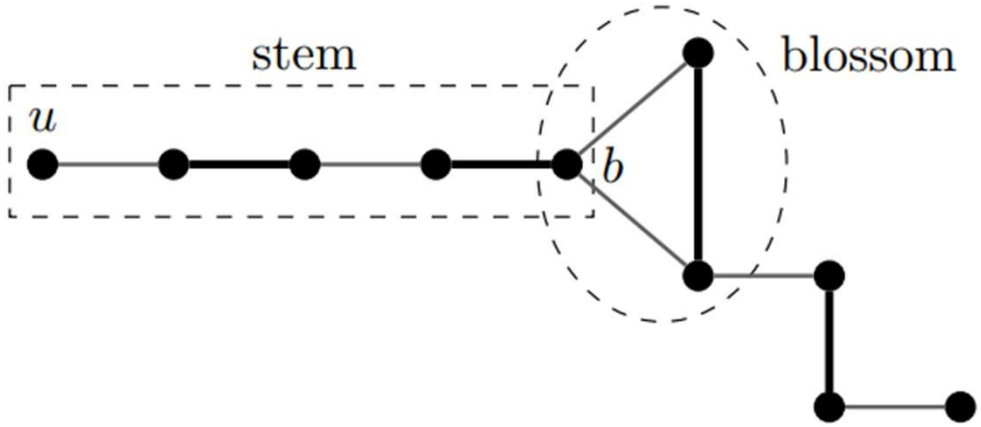
G_9

For example, we can find an alternating path of ***length 4*** from ***u*** to ***z***, namely ***u x y w z***, but another alternating path from ***u*** to ***z*** exists of ***length 3*** where the last edge is not matched, namely ***u x y z***. This is caused by the ***odd cycle occurring*** between y, w and z. Note that the vertex from which these two paths diverge (namely ***y***) is entered by a matched edge (***xy***) and has two possible unmatched edges out (***yw*** and ***yz***). This configuration is the basis behind the **blossom**.

Edmonds' Blossom Algorithm

Definition 5.17 Given a graph G and a matching M, a ***flower*** is the union of two M-alternating paths from an unsaturated vertex u to another vertex v where one path has odd length and the other has even length.

The ***stem*** of the flower is the maximal common initial path out of u, that ends at a vertex b, called the ***base***. The ***blossom*** is the odd cycle that is obtained by removing the stem from the flower.



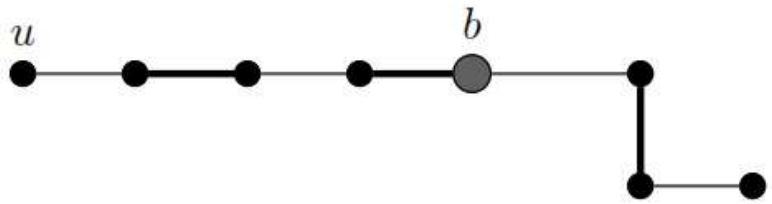
G_{10}

A few additional details can be discerned from these definitions. First, the stem must be of even length since the last edge must be from the matching M. Next, the blossom is an odd cycle C_{2k+1} where exactly k edges are from M, since the two edges from the base on the cycle must both be unmatched edges. Moreover, every vertex of the blossom must be saturated by M since traveling some direction along the cycle will end with an edge from M. Finally, the only edges that come off the blossom that are also from M must be from the stem.

Edmonds' Blossom Algorithm starts in the same way as the Augmenting Path Algorithm, where we examine alternating paths originating from an unsaturated vertex. Where these algorithms differ is when a blossom is encountered. When all alternating paths from u are being explored, if two different paths to a vertex are found to end in different types of edges (namely matched or unmatched), then a blossom has been discovered. We can then

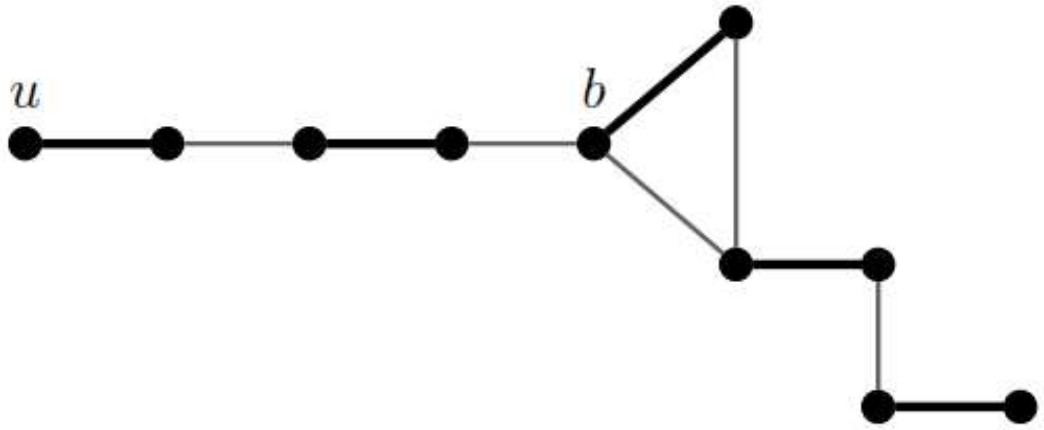
Point 1

contract the blossom, much in the same way we contract an edge from Chapter 4 in the proof of Menger's Theorem. The contraction of the blossom from G_{10} above is shown below.



G_{10} with blossom contracted

If we find an augmenting path in the contracted graph, then we can find an augmenting path in the original graph by choosing the correct direction along the blossom. Now just like in the Augmenting Path Algorithm we can swap edges along this path, creating a larger matching while still maintaining the properties of a matching.



G_{10} with matching swapped

Edmonds' Blossom Algorithm

A **blossom** is a set of nodes and edges starting at an open vertex, with a "stem" of even number of edges (matched and unmatched edges alternating), and a cycle of odd number of edges (again with alternating matched and unmatched edges, but with the two edges adjacent to the stem unmatched). A illustration is shown in Figure 8.1(a).

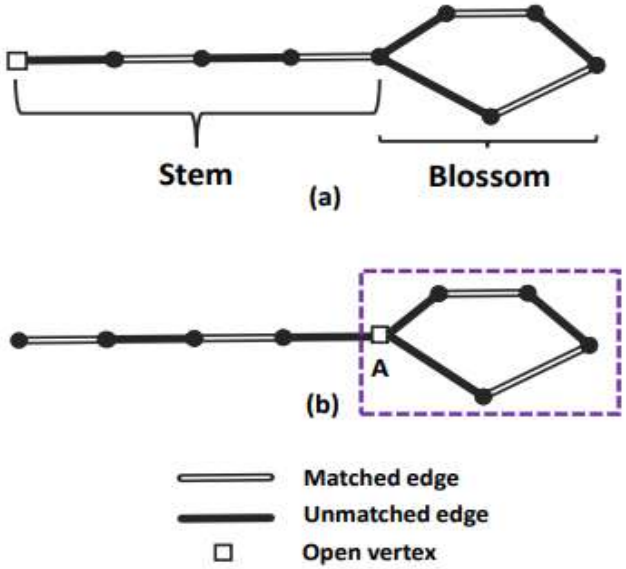
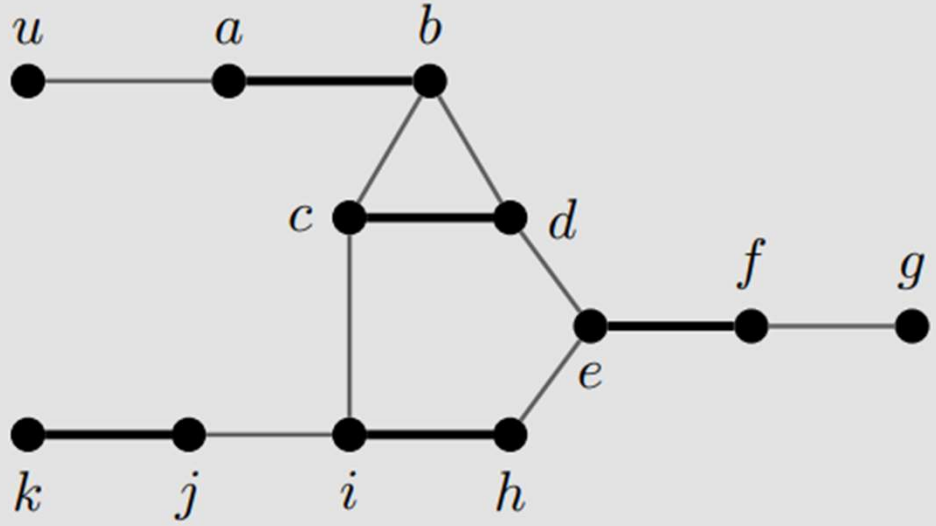


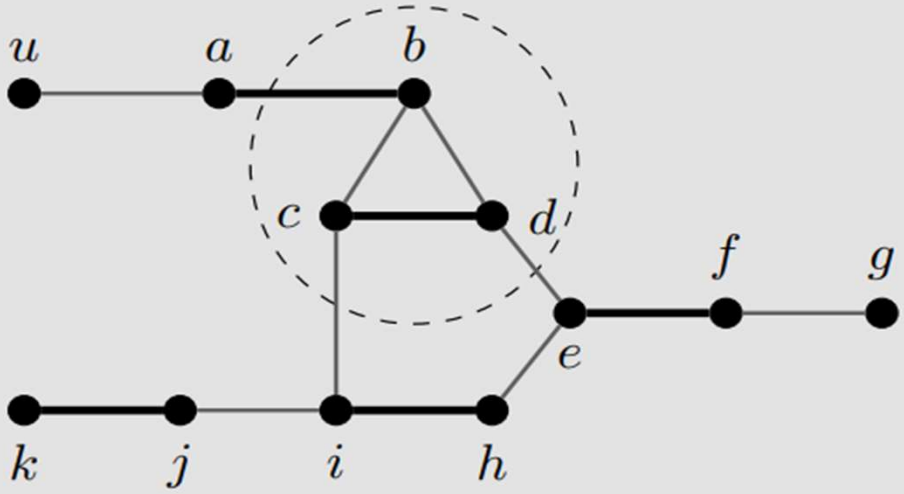
Figure 8.1: An example of blossom and the toggling of the stem.

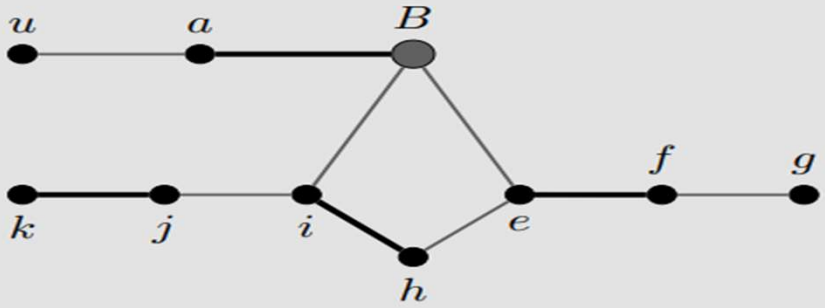
http://www.cs.cmu.edu/~anupamg/advalgos15/lectures/lecture08.pdf

Example 5.11 Use Edmonds' Blossom Algorithm to find a maximum matching on the graph below, where the initial matching is shown in bold.

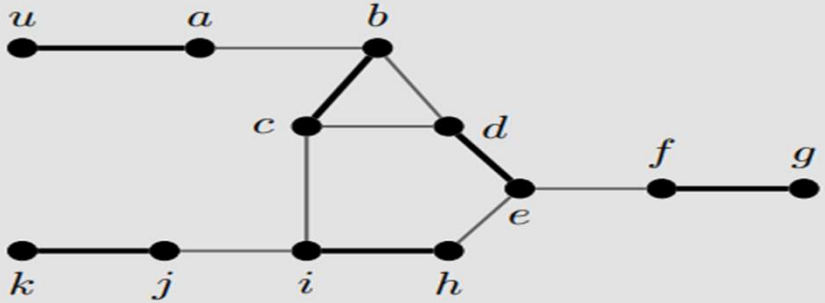


Solution: We begin by looking for alternating paths out of u. Note that we will explore all paths simultaneously (essentially building a Breadth-First Tree). At b we have two options for finding an alternating path, namely $u\,a\,b\,c$ and $u\,a\,b\,d$. If we take either path out to their next vertex, we get the ending vertex of the other previous path (such as $u\,a\,b\,d\,c$ and vertex c from the path $u\,a\,b\,c$). This implies that we have found a blossom, with odd cycle $b\,c\,d\,b$ and base vertex b. We contract this blossom to a vertex B as shown below.





Again we build out alternating paths from u. In doing so, we find the path $u\,a\,B\,e\,f\,g$, which is augmenting since it is alternating with both endpoints being unsaturated. Thus we retrace this path in the original graph as $u\,a\,b\,c\,d\,e\,f\,g$ and swap all edges along this path to obtain a larger matching.



The matching shown above is maximum since all vertices are saturated (and so is also a perfect matching).

Definition 5.18 A perfect matching is ***stable*** if no unmatched pair is ***unstable***; that is, if x and y are not matched but both rank the other higher than their current partner, then x and y form an unstable pair.

Stable Matching Problem

Given two sets $R = \{r_1, \ldots, r_n\}, H = \{h_1, \ldots, h_n\}$

each agent ranks **every** agent in the other set.

Stable Matching Problem

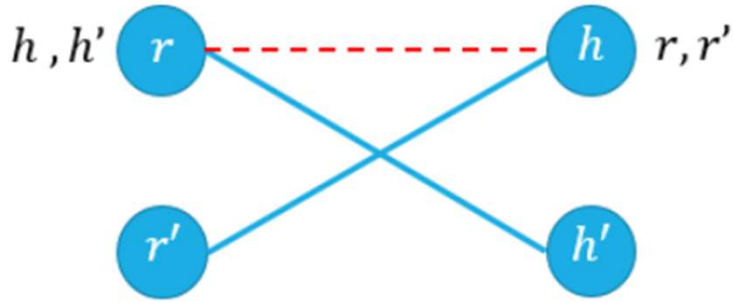
Given two sets $\mathrm{R} = \{r_1, \ldots, r_n\}, H = \{h_1, \ldots, h_n\}$

each agent ranks **every** agent in the other set.

Goal: Match each agent to **exactly one** agent in the other set, respecting their preferences.

How do we "respect preferences"?

Avoid blocking pairs: unmatched pairs (r, h) where r prefers h to their match, and h prefers r to its match.



Stable Matching, More Formally

Perfect matching:

- Each rider is paired with exactly one horse.
- Each horse is paired with exactly one rider.

Stability: no ability to exchange

an unmatched pair r-h is blocking if they both prefer each other to current matches.

Stable matching: perfect matching with no blocking pairs.

Algorithm Example

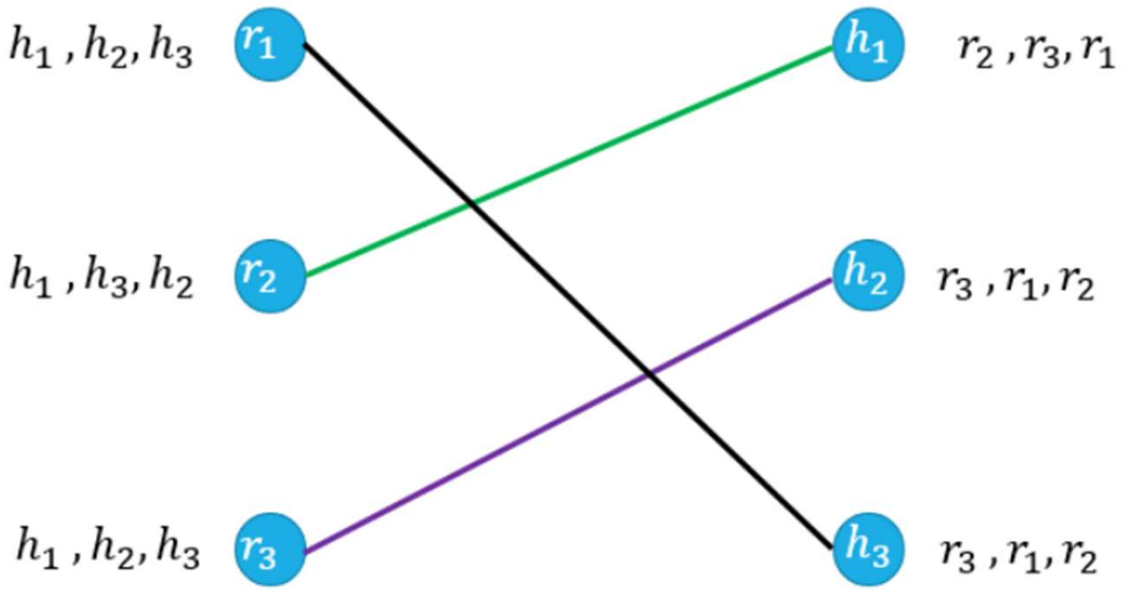
h_1, h_2, h_3 r_1 h_1 r_2, r_3, r_1

h_1, h_3, h_2 r_2 h_2 r_3, r_1, r_2

h_1, h_2, h_3 r_3 h_3 r_3, r_1, r_2

Proposals: $r_1, r_2, r_1, r_3, r_3, r_1$

Algorithm Example



Proposals: $r_1, r_2, r_1, r_3, r_3, r_1$

Gale-Shapley Algorithm

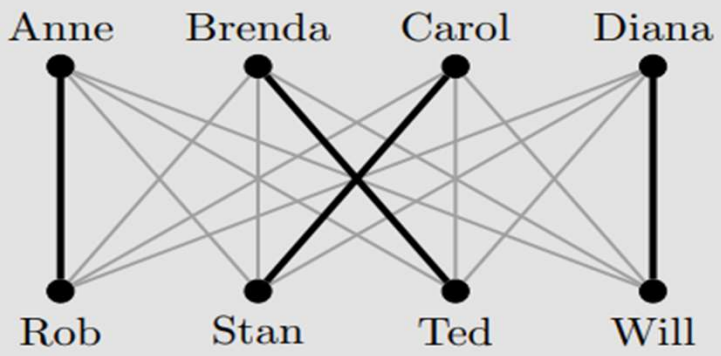
Initially all r in R and h in H are free
While there is a free r
    Let h be highest on r's list that r has not proposed to
    if h is free, then match (r, h)
    else //h is not free
    suppose (r', h) are matched
        if h prefers r to r'
            unmatch (r', h)
            match (r, h)

Example 5.13 Four men and four women are being paired into marriages. Each person has ranked the members of the opposite sex as shown below. Draw a bipartite graph and highlight the matching Anne–Rob, Brenda–Ted, Carol–Stan, and Diana–Will. Determine if this matching is stable. If not, find a stable matching and explain why no unstable pair exists.

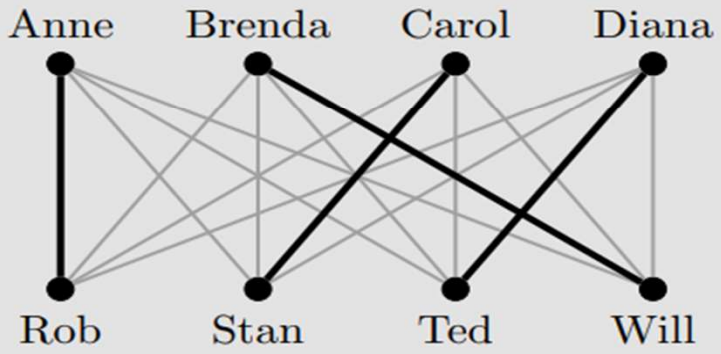
Anne:	t	>	r	>	s	>	w	Rob:	a	>	b	>	c	>	d
Brenda:	s	>	w	>	r	>	t	Stan:	a	>	c	>	b	>	d
Carol:	w	>	r	>	s	>	t	Ted:	c	>	d	>	a	>	b
Diana:	r	>	s	>	t	>	w	Will:	c	>	b	>	a	>	d

Solution: The complete bipartite graph appears on the next page with the matching in bold. This matching is not stable since Will and Brenda form an unstable pair, as they prefer each other to their current mate.

Anne:	t	>	r	>	s	>	w	Rob:	a	>	b	>	c	>	d
Brenda:	s	>	w	>	r	>	t	Stan:	a	>	c	>	b	>	d
Carol:	w	>	r	>	s	>	t	Ted:	c	>	d	>	a	>	b
Diana:	r	>	s	>	t	>	w	Will:	c	>	b	>	a	>	d

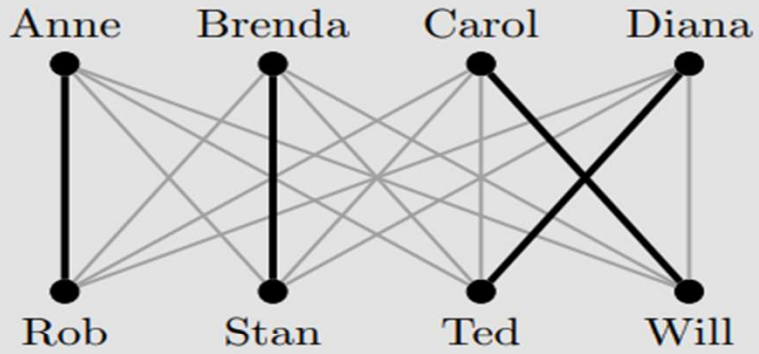


Switching the unstable pairs produces the matching below (Anne ↔ Rob, Brenda ↔ Will, Carol ↔ Stan, and Diana ↔ Ted).



Anne:	t	>	r	>	s	>	w		Rob:	a	>	b	>	c	>	d
Brenda:	s	>	w	>	r	>	t		Stan:	a	>	c	>	b	>	d
Carol:	w	>	r	>	s	>	t		Ted:	c	>	d	>	a	>	b
Diana:	r	>	s	>	t	>	w		Will:	c	>	b	>	a	>	d

Note that this matching is not stable either since Will and Carol prefer each other to their current mate. Switching the pairs produces a new matching (Anne ↔ Rob, Brenda ↔ Stan, Carol ↔ Will, Diana ↔ Ted) shown below.



This final matching is in fact stable due to the following: Will and Carol are paired, and each other's first choice; Anne only prefers Ted over Rob, but Ted does not prefer Anne over Diana; Brenda does not prefer anyone over Stan; Diana prefers either Rob or Stan over Ted, but neither of them prefers Diana to their current mate.

Gale-Shapley Algorithm

Input: Preference rankings of n women and n men.

Steps:

1. Each man proposes to the highest ranking woman on his list.
2. If every woman receives only one proposal, this matching is stable. Otherwise move to Step (3).
3. If a woman receives more than one proposal, she
   (a) accepts if it is from the man she prefers above all other currently available men and rejects the rest; or,
   (b) delays with a maybe to the highest ranked proposal and rejects the rest.
4. Each man now proposes to the highest ranking unmatched woman on his list who has not rejected him.
5. Repeat Steps (2)–(4) until all people have been paired.

Output: Stable Matching.

Instance of the Stable Marriage Problem

An instance of the stable marriage problem can be specified either by two sets of preference lists or by a ranking matrix, as in the example below.

men's preferences

	1st	2nd	3rd
Bob:	Lea	Ann	Sue
Jim:	Lea	Sue	Ann
Tom:	Sue	Lea	Ann

women's preferences

	1st	2nd	3rd
Ann:	Jim	Tom	Bob
Lea:	Tom	Bob	Jim
Sue:	Jim	Tom	Bob

ranking matrix

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

{(Bob, Ann) (Jim, Lea) (Tom, Sue)} is unstable

{(Bob, Ann) (Jim, Sue) (Tom, Lea)} is stable

Stable Marriage Algorithm (Gale-Shapley)

Step 0 Start with all the men and women being free

Step 1 While there are free men, arbitrarily select one of them and do the following:

Proposal The selected free man *m* proposes to *w*, the next woman on his preference list

Response If *w* is free, she accepts the proposal to be matched with *m*. If she is not free, she compares *m* with her current mate. If she prefers *m* to him, she accepts *m*'s proposal, making her former mate free; otherwise, she simply rejects *m*'s proposal, leaving *m* free

Step 2 Return the set of *n* matched pairs

Example

Free men:
Bob, Jim, Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Bob proposed to Lea
Lea accepted

Free men:
Jim, Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Jim proposed to Lea
Lea rejected

Free men:
Jim, Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Jim proposed to Sue
Sue accepted

Free men:
Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Tom proposed to Sue
Sue rejected

Free men:
Tom

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Tom proposed to Lea
Lea replaced Bob
with Tom

Free men:
Bob

	Ann	Lea	Sue
Bob	2,3	1,2	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Bob proposed to Ann
Ann accepted

Analysis of the Gale-Shapley Algorithm

- The algorithm terminates after no more than n^2 iterations with a stable marriage output
- The stable matching produced by the algorithm is always *man-optimal*: each man gets the highest rank woman on his list under any stable marriage. One can obtain the *woman-optimal* matching by making women propose to men
- A man (woman) optimal matching is unique for a given set of participant preferences
- The stable marriage problem has practical applications such as matching medical-school graduates with hospitals for residency training

Practice

EX #:4.6

Problems: 4.1-4.6, 4.9, 4.10, 4.13, 4.14, 4.15,4.17, 4.20.

EX #: 5.5

Problems: 5.1-5.14, 5.17-5.19, 5.24, 5.25