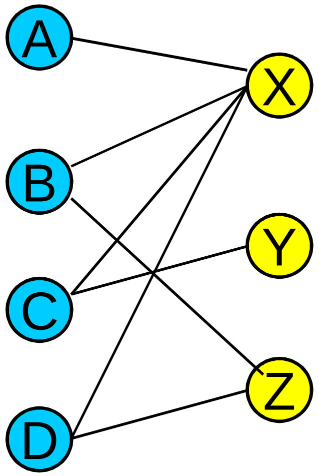
BIPARTITE MATCHING

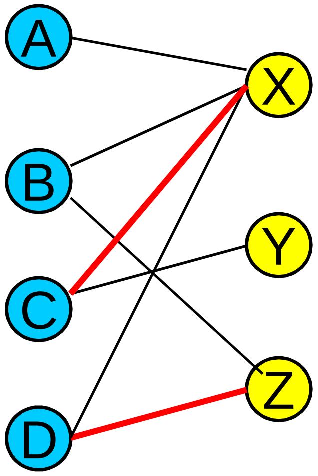
Bipartite Matching

- Example
  - given a community with *n* men and *m* women
- Assume we have a way to determine which couples (man/woman) are compatible for marriage
  - E.g. (Joe, Susan) or (Fred, Susan) but not (Frank, Susan)
- Problem: Maximize the number of marriages
  - Neither polygamy nor polyandry are allowed.. Each vertex can be assigned only one vertex of the other group.
  - solve this problem by creating a flow network out of a bipartite graph

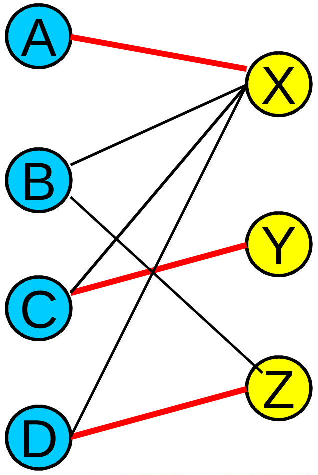
A Bipartite Graph

- It is an undirected graph $G=(V, E)$ in which V can be partitioned into **two (disjoint) sets** V_1 and V_2 such that $(u, v) \in E$ implies either $u \in V_1$ and $v \in V_2$ or vice versa.
- That is, all edges go between the two sets V_1 and V_2 but are not allowed within $\boldsymbol{V_1}$**(blue)** and $\boldsymbol{V_2}$**(yellow)**.
- A bipartite graph is *2-colorable*: the vertices can be colored in two colors so that every edge has its vertices colored differently.





Matching between x and c, and z and d.



Optimal matching

Bipartite graph: a graph whose vertices can be partitioned into two disjoint sets V and U, not necessarily of the same size, so that every edge connects a vertex in V to a vertex in U.

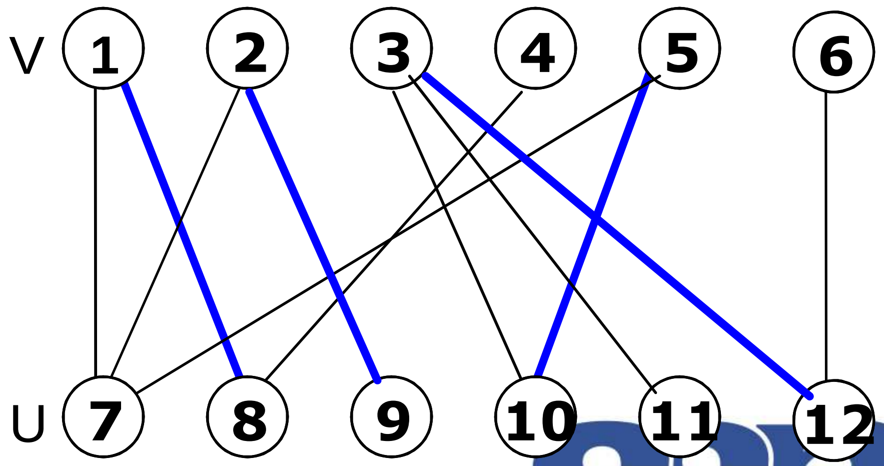
A graph is bipartite if and only if it does not have a cycle of an odd length. 2-Colorable, iff contains no odd length cycle.

Suppose V and U are opposite set of genders..

A *matching* in a graph is a subset of its edges with the property that no two edges share a vertex.

A *maximum* (or *maximum cardinality*) *matching* is a matching with the largest number of edges

- always exists
- not always unique



- Algorithm: Successive improvements, like Dijkstra’s algorithm (relaxation).
  - Suppose M is an initial matching, $M \subset G$ and $M \subset M'$. Iterate to improve M to M’, if an augmentation path is feasible, if not then $M=M' \subseteq G$.

  - Search for such an augmentation path: Starting with an unmatched partner $V \notin M$ of the smaller set, and similarly ending at an unmatched partner such that every second edge $\in M$, but the initial $E \notin M$. An *alternating* path, relative to the *M*, because it alternates the current edges to the edges not in the current matching M, therefore *augmenting-path* with respect to M.

  - *augmenting-path a*lways provides an improvement in the # of total matches, since unmatched are additionally matched.

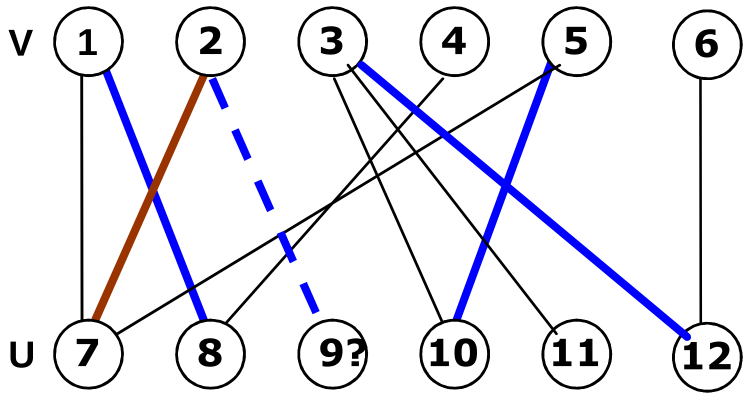
  - An augmenting path of a bipartite graph must have an add length path. Since all odd-numbered edges are $E_{2i+1} \notin M$, while all even-numbered ones are $E_{2i} \in M$, we increase the number of E (matching vertices) by one if replace the edge of $E_{2i} \in M$ with the edge of $E_{2i+1} \notin M$, where $0 \le i \le n$. The result is an improved M’, $|M'|=|M|+1$ which is odd numbered and also proof the Lemma to be given...

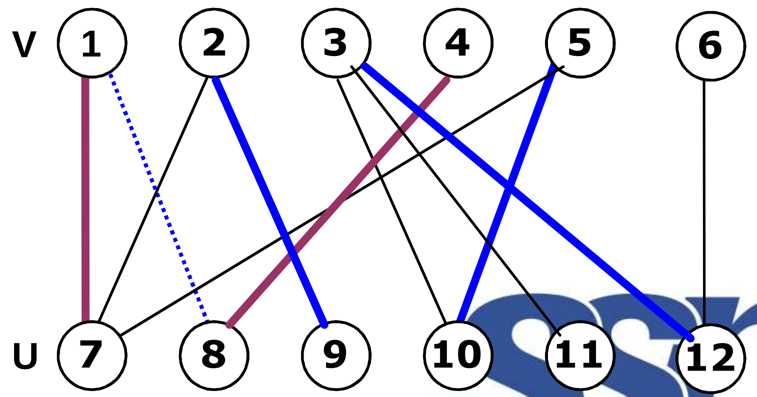
Lemma 1. Suppose that *G=(V,E)* is a bipartite graph with vertex classes *X* and *Y*, and *M* is a matching in *G*. If there exists an augmenting path from an unmatched vertex *x* in *X* to an unmatched vertex *y* in *Y*, then there exists a matching *M′* with cardinality |*M*| + 1.

Example below, if AP search would have started from the unmatched vertex 7, and proceed to 7-2, it would not have been resulted with an unmatched vertex having an only one path, and removal of the match is going leave it completely isolated, which could not be acceptable unless the number of vertices |U|> |V|. Besides that, search must end at the opposite set with an unmatched vertex.

This means, the maximum match is a perfect match which happens when |U|=|V|. Start from an unmatched but having a single path, either of the vertices 11, 4, 6.

- **<u>Theorem</u> A matching M is maximum if and only if there exists no augmenting path with respect to M.**





Augmenting Path Method (template)

- Start with some initial matching
  - e.g., the empty set
- Find an augmenting path and augment the current matching along that path
  - e.g., using bfs like traversal starting simultaneously from all the free vertices of the smaller set.
- AP is odd in length. When no augmenting path can be found, terminate and return the last matching which must be the case of maximum cardinality.

BFS-based Augmenting Path Algorithm

- Initialize queue Q with all free vertices in one of the sets (say V)

- While Q is not empty, delete front vertex *w* and label every unlabeled vertex *u* adjacent to *w* as follows:

  Case 1 (*w* is in V)

  If *u* is free, augment the matching along the path ending at *u* by moving backwards until a free vertex in V is reached. After that, erase all labels and reinitialize Q with all the vertices in V that are still free

  If *u* is matched (not with *w*), label *u* with *w* and enqueue *u*

  Case 2 (*w* is in U) Label its matching mate *v* with *w* and enqueue *v*

- After Q becomes empty, return the last matching, which is maximum.

Aggregate pairwise happiness of collective marriage.. $\Sigma(c_i)$..

Best collective coupling $y' = \text{argmax}_y(\Sigma(c_i))$

Maximum-weight perfect bipartite match (assignment problem, exactly solvable in $O(n^3)$ Hungarian algorithm..

Relaxation: We know the scores..

Measure edge features..

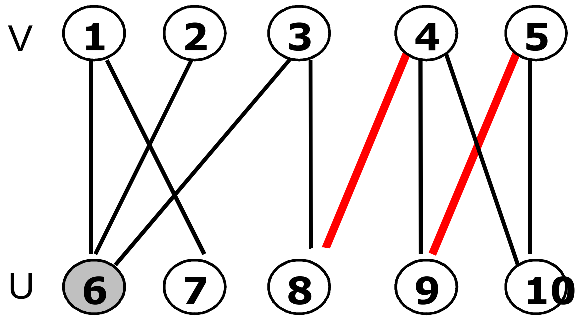
Complex feature vectors. Then instead of parameterizing features use MLE method..

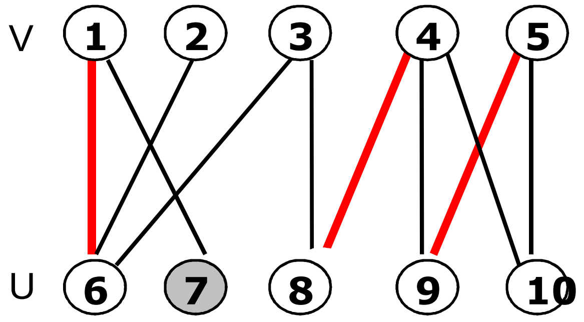
$c_i = f(x_i : \theta)$

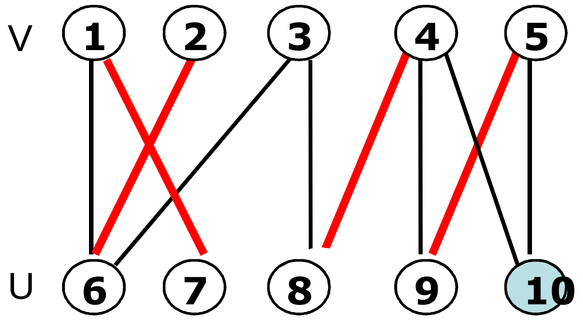
Probability of a match given a g, $p(y|x; \theta) = e^{(<\phi(x, y)> - g(x, \theta))}$ where y is a match for graph x.. Most likely match $y' = \text{argmax}_y(\phi(x, y), \theta)$ idea to construct data $(\phi(x, y), \theta) = \Sigma_i(c_{iy(i)})$ which suggests that $\phi(x, y) = \Sigma_i(\psi_{iy(i)})$

$c_{iy(i)} = <\psi_{iy(i)}, \theta>$

- Augmentation along path 1, 2
- Augmentation along path 2,6,1,7
- Augmentation along the paths 3, 8, 4, 9, 5, 10





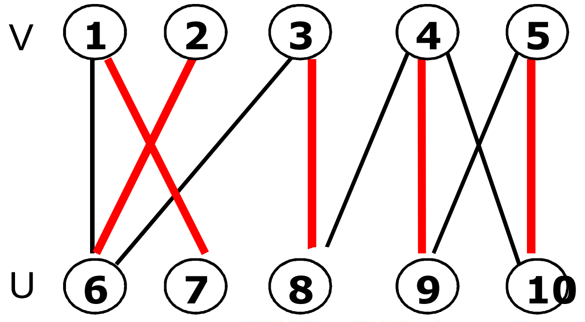


Queue: 1 2 3,
AP from 6

Queue: 2 3

Queue: 2 3 6 8 1 4
AP from 7

Queue: 3,
AP from 10
Queue: 3 6 8 2 4 9



Maximum match, no remaining vertices to be matched

Maximum-matching algorithm for bipartite graphs

```
Input: A bipartite graph $G = \langle V, U, E \rangle$
Output: A maximum-cardinality matching $M$ in the input graph
initialize set $M$ of edges with some valid matching (e.g., the empty set)
initialize queue $Q$ with all the free vertices in $V$ (in any order)
while not $Empty(Q)$ do
    $w \leftarrow Front(Q)$;  $Dequeue(Q)$
    if $w \in V$
        for every vertex $u$ adjacent to $w$ do
            if $u$ is free
                //augment
                $M \leftarrow M \cup (w, u)$
                $v \leftarrow w$
                while $v$ is labeled do
                    $u \leftarrow$ vertex indicated by $v$'s label;  $M \leftarrow M - (v, u)$
                    $v \leftarrow$ vertex indicated by $u$'s label;  $M \leftarrow M \cup (v, u)$
                remove all vertex labels
                reinitialize $Q$ with all free vertices in $V$
                break //exit the for loop
            else //$u$ is matched
                if $(w, u) \notin M$ and $u$ is unlabeled
                    label $u$ with $w$
                    $Enqueue(Q, u)$
    else //$w \in U$ (and matched)
        label the mate $v$ of $w$ with "$w$"
        $Enqueue(Q, v)$
return $M$ //current matching is maximum
```

Maximum Matching Algorithm Analysis

- Each iteration (except the last) matches two free vertices (one each from V and U). Therefore, the number of iterations cannot exceed $\lfloor n/2 \rfloor + 1$, where n is the number of vertices in the graph. The time spent on each iteration is in $O(n+m)$, where m is the number of edges in the graph. Hence, the time efficiency is in $O(n(n+m))$

- This can be improved to $O(n(n+m))$ by combining multiple iterations to maximize the number of edges added to matching M in each search..

- Finding a maximum matching in an arbitrary graph is much more difficult, but the problem was solved in 1965 by Jack Edmonds.

Stable Marriage Problem

- There is a set Y = $\{m_1,\ldots,m_n\}$ of n men and a set X = $\{w_1,\ldots,w_n\}$ of n women. Each one has a ranking list of the opposite gender, (with no ties in these lists).
- A *marriage matching* M is a set of n pairs (m_i, w_j).
- A pair (m, w) is said to be a *blocking pair* for matching M if man m and woman w are not matched in M but prefer each other to their mates in M.
- A marriage matching M is called *stable* if there is no blocking pair for it; otherwise, it’s called *unstable*.
- The *stable marriage problem* is to find a stable marriage matching for men’s and women’s given preferences.

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.

<u>men's preferences</u>

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

<u>women's preferences</u>

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

<u>ranking matrix</u>

	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 because even if the algorithm is iterated once more the result will not change (due to the preferences of the proposed parties)!!.. Algorithm ends when the stability is achieved. In which case there is no a blocking pair.

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	$\boxed{1,2}$	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Free men:
Jim, Tom

	Ann	Lea	Sue
Bob	2,3	$\boxed{1,2}$	3,3
Jim	3,1	1,3	2,1
Tom	3,2	2,1	1,2

Free men:
Tom

Bob proposed to Lea
Lea accepted

Jim proposed to Lea
Lea rejected

Jim proposed to Sue

Sue accepted

Tom proposed to Sue

Sue rejected

Tom proposed to Lea
Lea replaced Bob with Tom

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

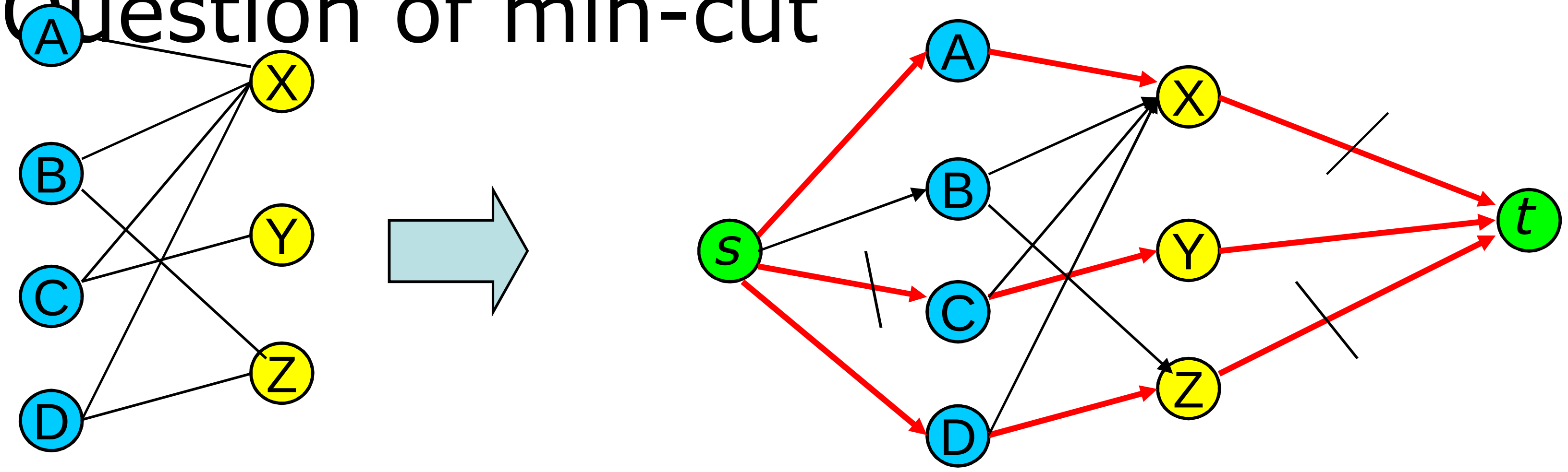
- The stable matching produced by the algorithm is always *gender-optimal*: each man gets the highest rank woman on his list under any stable marriage. One can obtain the *woman-optimal* matching by making women propose to men

- 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

Solution Using Max Flow

- Add a supersource, supersink, make each undirected edge directed with a flow of 1
- Question of min-cut



Since the input is 1, flow conservation prevents multiple matchings