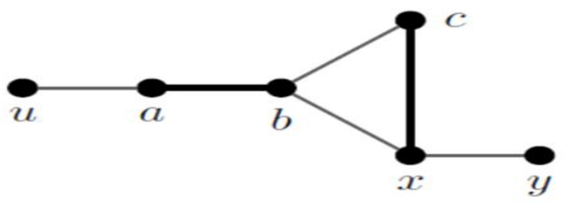
MATCHING IN GENERAL GRAPHS

- We have only discussed matchings inside of bipartite graphs, although Berge's Theorem was stated to hold for non-bipartite graphs as well.
- Finding matchings in general graphs is often more complex than in bipartite graphs.
- Berge's Theorem holds (M is maximum if and only if G has no M-augmenting paths) yet the Augmenting Path Algorithm only works on bipartite graphs.

Intuitive Idea:

The main sticking point is that in finding alternating paths from an unsaturated vertex, there could be more than one path to x, and only investigating one would miss an augmenting path.

For example, if we are searching for alternating paths from u to x in graph G_7 below, we might choose u a b x. If instead, we chose u a b c x, we could continue the alternating path to find an augmenting path to y (namely, u a b c x y).

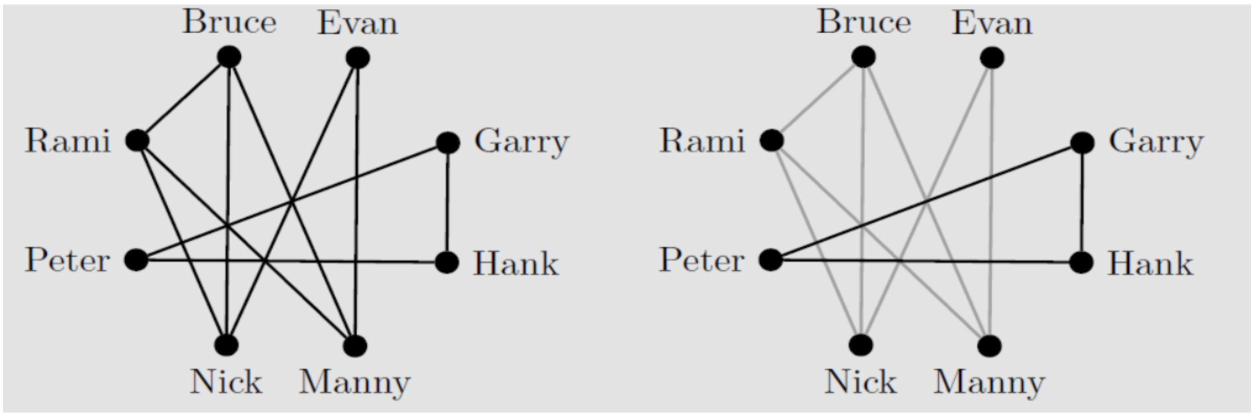


Jack Edmonds devised a modification to the Augmenting Path Algorithm that works on general graphs, accounting for these odd cycles [28].

This algorithm is more powerful than we need for small examples, as these can usually be determined through inspection and some reasoning.

Example 5.10 Halfway through the canoe trip from Example 5.9, Rami will no longer share a canoe with Garry, and Hank angered Evan so they cannot share a canoe. Update the graph model and determine if it is now possible to pair the eight men into four canoes.

Solution: The updated graph only removes two edges, but in doing so the graph is now disconnected. A disconnected graph could have a perfect matching so long as each component itself has a perfect matching.



In this case, a perfect matching is impossible since Peter, Garry, and Hank form one component and so at most two of them can be paired. Likewise, the other component contains five people and so at most four can be paired. Thus there is no way to pair the eight men into four canoes.

The theorem below is one of the classic matching theorems for general graphs, due to the British mathematician William Tutte, who made significant contributions to graph theory after World War II, where he served as a code breaker.

- The result relates the number of vertices removed from a graph to the number of odd components created with their removal, similar to Hall's Theorem relating the size of a set with the size of its neighbor set.

We will use the following notation throughout the proof:

Definition 5.14 Let G be a graph. Define $o(G)$ to be the number of odd components of G, that is the number of components containing an odd number of vertices.

In Example 5.10 above, σ(G) = 2 since it contains two components, one with 5 vertices (with Bruce, Evan, Manny, Nick, and Rami) and the other with 3 (Gary, Hank, and Peter).

Theorem 5.15 (Tutte's Theorem) A graph G has a perfect matching if and only if for every proper subset of vertices S the number of odd components of $G - S$ is at most $|S|$, that is $o(G - S) \leq |S|$.

Proof: For ease of use, we will refer to the following statement as Tutte's condition throughout this proof:

$$o(G-S) \leq |S| \text{ for every } S \subsetneq V(G)$$

First, suppose G has a perfect matching and S is any subset of vertices from G. Then every odd component of $G-S$ must have an edge to a vertex from S (since every vertex must be part of the matching). Thus the number of vertices in S must be at least the number of odd components of $G-S$, that is G satisfies Tutte's condition.

Conversely, assume $|G| = n$. We will show that G has a perfect matching whenever it satisfies Tutte's condition. First note that if $S = \emptyset$, then $|S| = 0$ and $G - S = G$. Thus

$$o(G) = o(G-S) \leq |S| = 0$$

and so G must only have even components. Thus n must be even. Moreover, for every vertex we add to S, we will affect the size of exactly one component from $G-S$, and so $|S|$ and $o(G-S)$ must have the same parity.

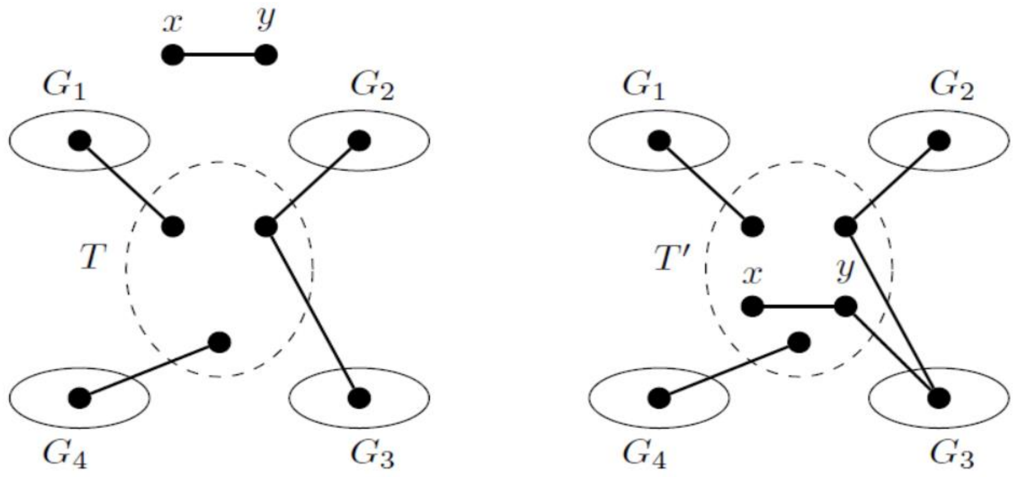
We will argue by induction on n (where n must always be an even integer) that if G satisfies Tutte's condition then G will have a perfect matching.

First suppose $n = 2$. Since G cannot have any odd components, we know $G = K_2$. Thus G has a perfect matching, namely the single edge in G.

Now suppose for some $n \geq 4$ that all graphs H of even order $n' < n$ satisfying Tutte's condition have a perfect matching. Let G be a graph of order n satisfying Tutte's condition. We will consider whether $o(G-S) < |S|$ for all proper subsets S or if there exists some $S \subsetneq V(G)$ where $o(G-S) = |S|$. Note we only need to consider sets S with $2 \leq S \leq n$.

Case 1: Suppose $o(G-S) < |S|$ for all $S \subsetneq V(G)$. Since $o(G-S)$ and $|S|$ are of the same parity, we know $o(G-S) \leq |S| - 2$ for all $S \subsetneq V(G)$ with $2 \leq |S| \leq n$. Let $e = xy$ be some edge of G and let $G' = G - \{x, y\}$.

Let $T \subsetneq V(G')$ and let $T' = T \cup \{x, y\}$. Then $|T'| = |T| + 2$.



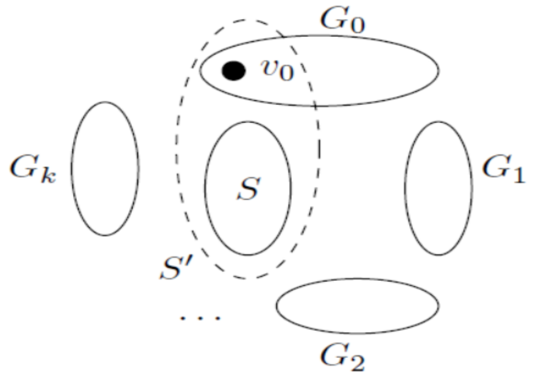
If $o(G' - T) > |T|$ then $o(G' - T) > |T'| - 2$. Note that $o(G' - T) = o(G - T')$ and so $o(G - T') \geq |T'|$, which is a contradiction to the fact that G satisfies Tutte's condition. Thus $o(G' - T) \leq |T|$, and so by the induction hypothesis G' has a perfect matching M. Together with the edge $e = xy$, $M \cup e$ is a perfect matching of G.

Case 2: Suppose there exists some subset $S' \subsetneq V(G)$ satisfying

$$(*)\ o(G - S') = |S'|$$

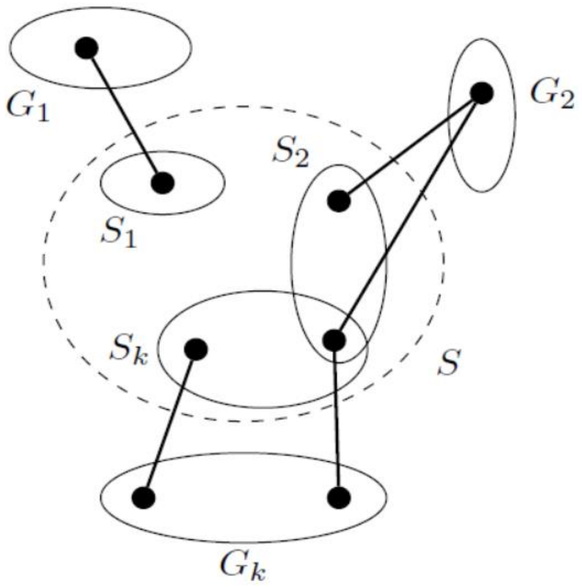
Among all possible sets S', pick S to be the largest possible one satisfying condition $(*)$, say $|S| = k$.

Let $G_1, G_2, \ldots, G_k$ be the odd components of $G - S$. Suppose G_0 were some even component of $G - S$, and let $v_0 \in V(G_0)$. Then $G_0 - \{v_0\}$ would consist of at least one odd component. Let $S' = S \cup \{v_0\}$.



Thus $o(G - S') \geq k + 1 = |S'|$. But then $o(G - S') = |S'|$ since G satisfies Tutte's condition, and so S' is a larger set than S satisfying $(*)$, a contradiction to our choice of S. Thus $G - S$ can only consist of odd components.

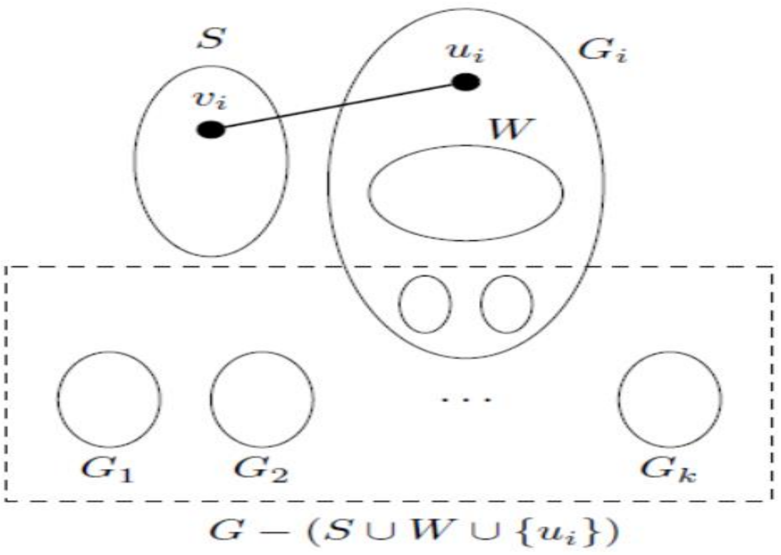
For $i = 1, \ldots, k$ let S_i denote the vertices of S adjacent to at least one vertex of G_i. Since G does not contain any odd components, we know that $S_i \neq \emptyset$ for all i.



Moreover, let R be the union of j of the S_i's. Then $R \subseteq S$ and in $G-R$ the only odd components that remain are the G_i for which $S_i \subseteq R$. Thus $o(G-R) = j$ and since $o(G-R) \leq |R|$, we know $|R| \geq j$. Then by Theorem 5.13 we can find a system of distinct representatives for $S_1, \ldots, S_k$; that is, there exists distinct vertices $v_1, \ldots, v_k$ so that $v_i \in S_i$ and for some $u_i \in G_i$ we have $v_i \sim u_i$.

We will show each $G_i - u_i$ has a perfect matching, which we can combine together with $v_i u_i$ to get a perfect matching of G. To do this, we simply must show that $G_i - u_i$ satisfies Tutte's condition. Note that since each G_i is an odd component, we know that $G_i - u_i$ has an even number of vertices.

For any i, consider $W \subsetneq V(G_i - u_i)$. If $o(G_i - u_i - W) > |W|$ then we know $o(G_i - u_i - W) \geq |W| + 2$. But then the odd components in $G - (S \cup W \cup \{u_i\})$ are exactly the odd components of G, except G_i, along with the odd components of $G_i - u_i - W$.



This implies

$$\begin{aligned} o(G - (S \cup W \cup \{u_i\})) &= o(G_i - u_i - W) + o(G - S) - 1 \\ &\geq |S| + |W| + 1 \\ &= |S \cup W \cup \{u_i\}| \end{aligned}$$

But this contradicts the maximality of S. Thus for each i we know $o(G_i - u_i - W) \leq |W|$, that is $G_i - u_i$ satisfies Tutte's condition. Thus by the induction hypothesis applied to every $G_i - u_i$ we know that $G_i - u_i$ has a perfect matching. Since $|S| = k$, $v_1u_1, v_2u_2, \ldots, v_ku_k$ forms a matching that saturates S and together with each of the perfect matchings of the $G_i - u_i$ we have a prefect matching of G.

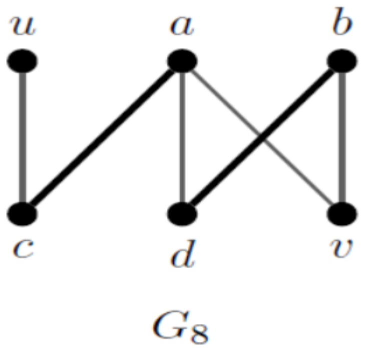
The following result can be seen as a corollary to Tutte's Theorem, as quick proof uses a counting argument between the degrees of each vertex and the number of odd components (see Exercise 5.23).

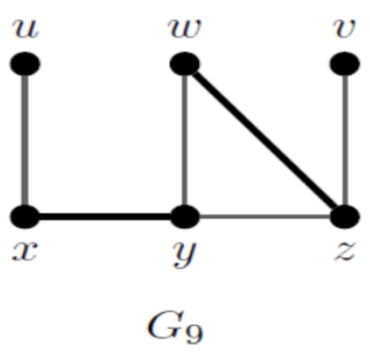
Edmonds' Blossom Algorithm:

- Jack Edmonds devised this procedure so that augmenting paths can be found within a non-bipartite graph, which can then be modified to create a larger matching.

Intuitive Idea:

To get a better understanding of the complexities that arise when searching for matching in a non-bipartite graph, consider the two graphs G_8 and G_9 shown below, where only the graph on the left is bipartite. In both graphs, the vertex u is left unsaturated by the matching is shown in bold.





Notice that in G_8 above, when looking for an alternating path out of u, if the end of the path is in the same partite set as u, then this path must have even length and the last edge of the path would be from the matching.

- This implies if an alternating path from u to a saturated vertex s has even length, then all paths from u to s have even length, and they must all use as their last edge the only matched edge out of s.
- However, in the non-bipartite graph on the right, this is not the case.

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.

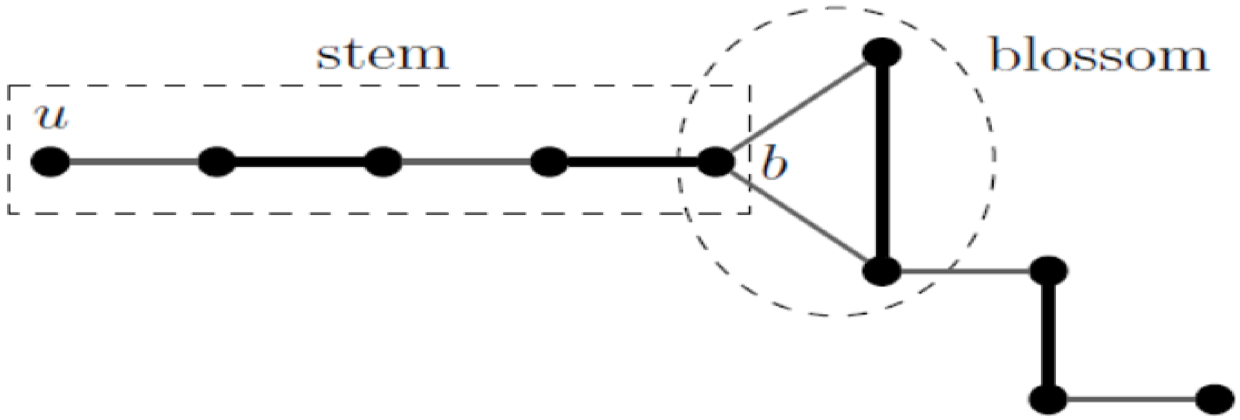
- Note that the vertex from which these two paths diverge (namely y) is entered by a matched edge (xy) and has two possible unmatched edges out (yw and yz).

Key point: **This configuration is the basis behind the blossom.**

Flower, Stem, Base and Blossom:

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}

Basic Detail:

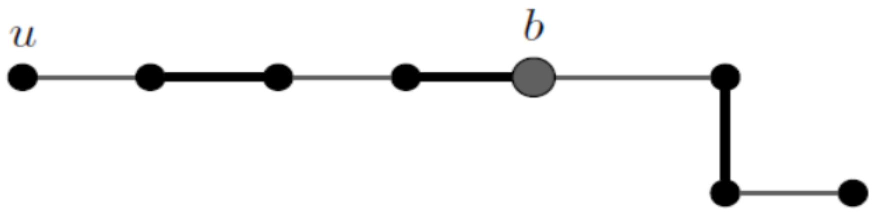
- The stem must be of even length since the last edge must be from the matching M.
- The blossom is an odd cycle C_{2k+1} where exactly k edges are from M, since the two edges from the base on the cycle must both be unmatched edges.
- Every vertex of the blossom must be saturated by M, since traveling some direction along the cycle will end with an edge from M .
- The only edges that come off the blossom that are also from M must be from the stem.
- Any non-stem edges coming off the blossom must be unmatched.

Edmonds' Blossom Algorithm starts in the same way as the Augmenting Path Algorithm.

Basic Steps for Algorithm:

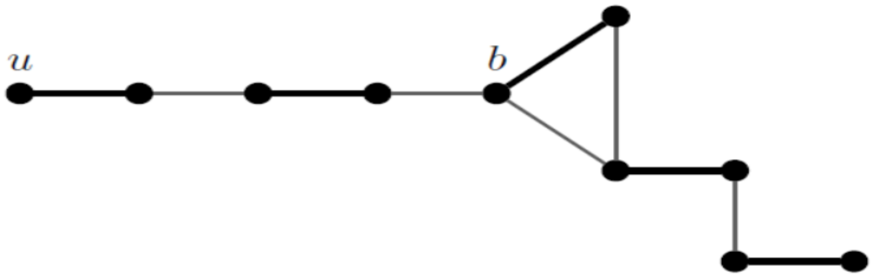
- We examine alternating paths originating from an unsaturated vertex.

- 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 contract the blossom, much in the same way we contract an edge from in the proof of Menger's Theorem.
- The contraction of the blossom from $G1_0$ 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.
- 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.

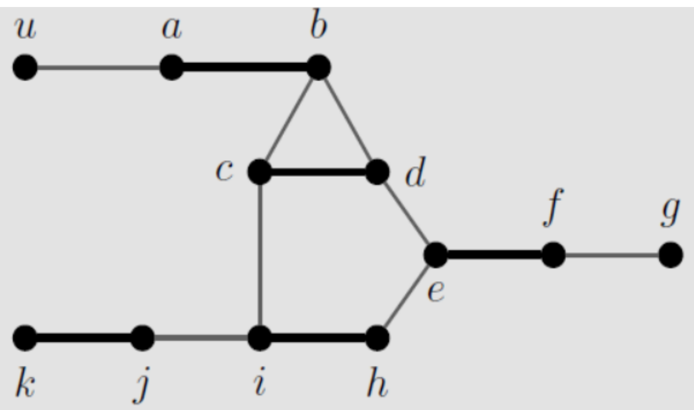


G_{10} with matching swapped

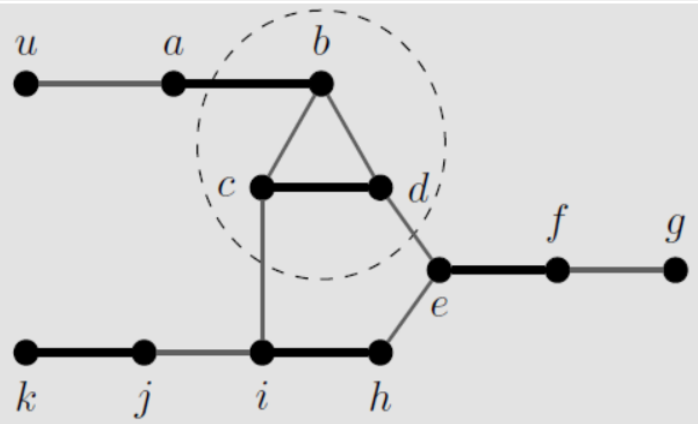
- The pseudocode for Edmonds' Algorithm appears in Appendix E.

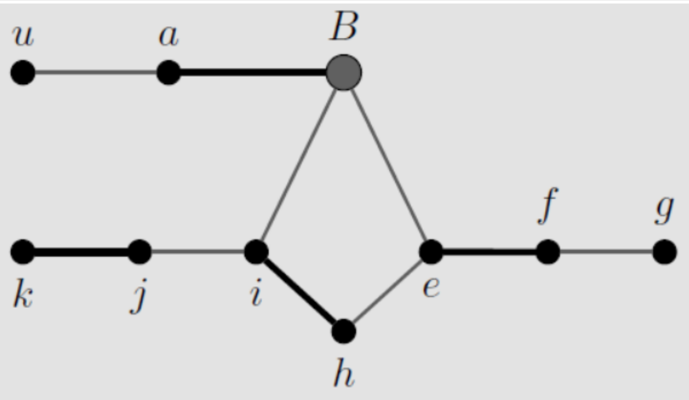
Example for Edmond's Algorithm:

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

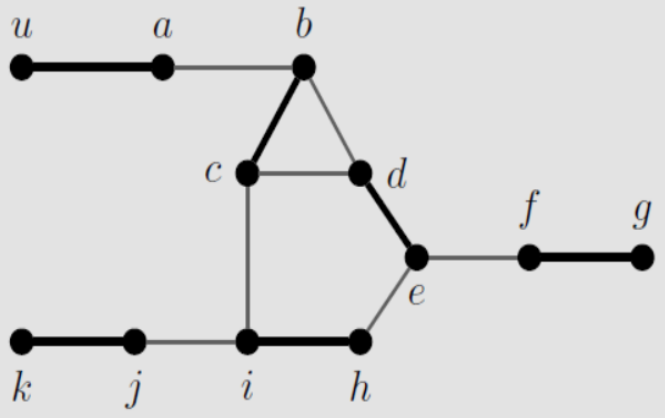


Solution: We begin by looking for alternating paths out of u. Note that we will explore all paths simultaneously (essentially building a Breadth-First Tree). At b we have two options for finding an alternating path, namely $u\,a\,b\,c$ and $u\,a\,b\,d$. If we take either path out to their next vertex, 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).

STABLE MATCHING:

- In our discussion of matchings so far, we have only been concerned with finding the largest matching possible; but in many circumstances, one pairing may be preferable over another.
- When taking preferences into account, we no longer focus on whether two items can be paired but rather which pairing is best for the system.
- Initially, we will only consider those situations that can be modeled by a bipartite graph with an equal number of items in X and Y .
- The previous models demonstrated undesirable pairings by leaving off the edge in the graph; but with preferences added, we include every edge
- possible in the bipartite graph.
- This means the underlying graph will be a complete bipartite graph.

The preference model for matchings is often referred to as the Stable Marriage Problem to parallel Hall's Marriage Theorem, and our terminology will reflect the marriage model.

Problem: We start with two distinct, yet equal-sized, groups of people, usually men and women, who have ranked the members of the other group.

Task: The stability of a matching is based on if switching two matched edges would result in happier couples.

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.

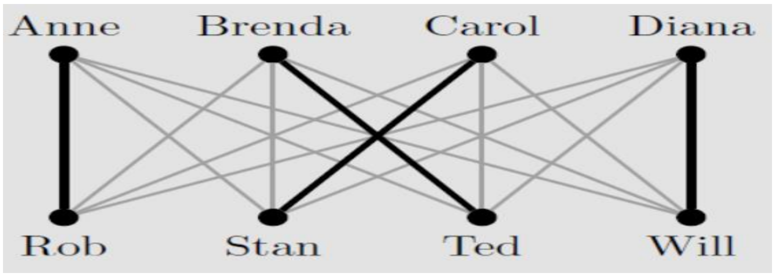
Remark: In essence, when pairing couples into marriages we want to ensure no one will leave their current partner for someone else.

- ✓ To better understand stability, consider the following example.

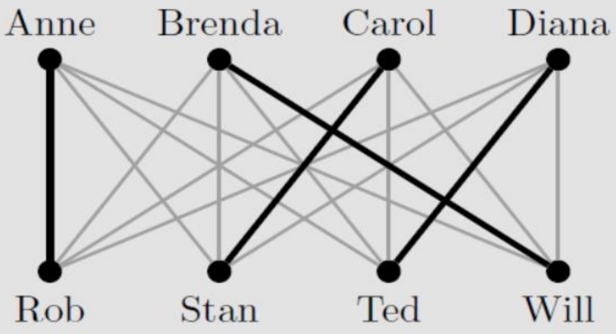
Example 5.13 Four men and four women are being paired into marriages. Each person has ranked the members of the opposite sex as shown below. Draw a bipartite graph and highlight the matching Anne–Rob, Brenda–Ted, Carol–Stan, and Diana–Will. Determine if this matching is stable. If not, find a stable 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

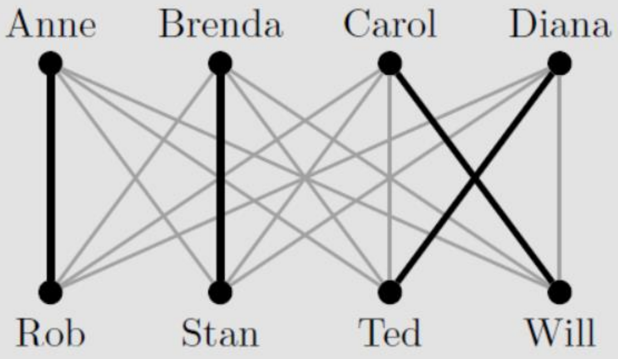
Solution: The complete bipartite graph appears on the next page with the matching in bold. This matching is not stable since Will and Brenda form an unstable pair, as they prefer each other to their current mate.



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



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.

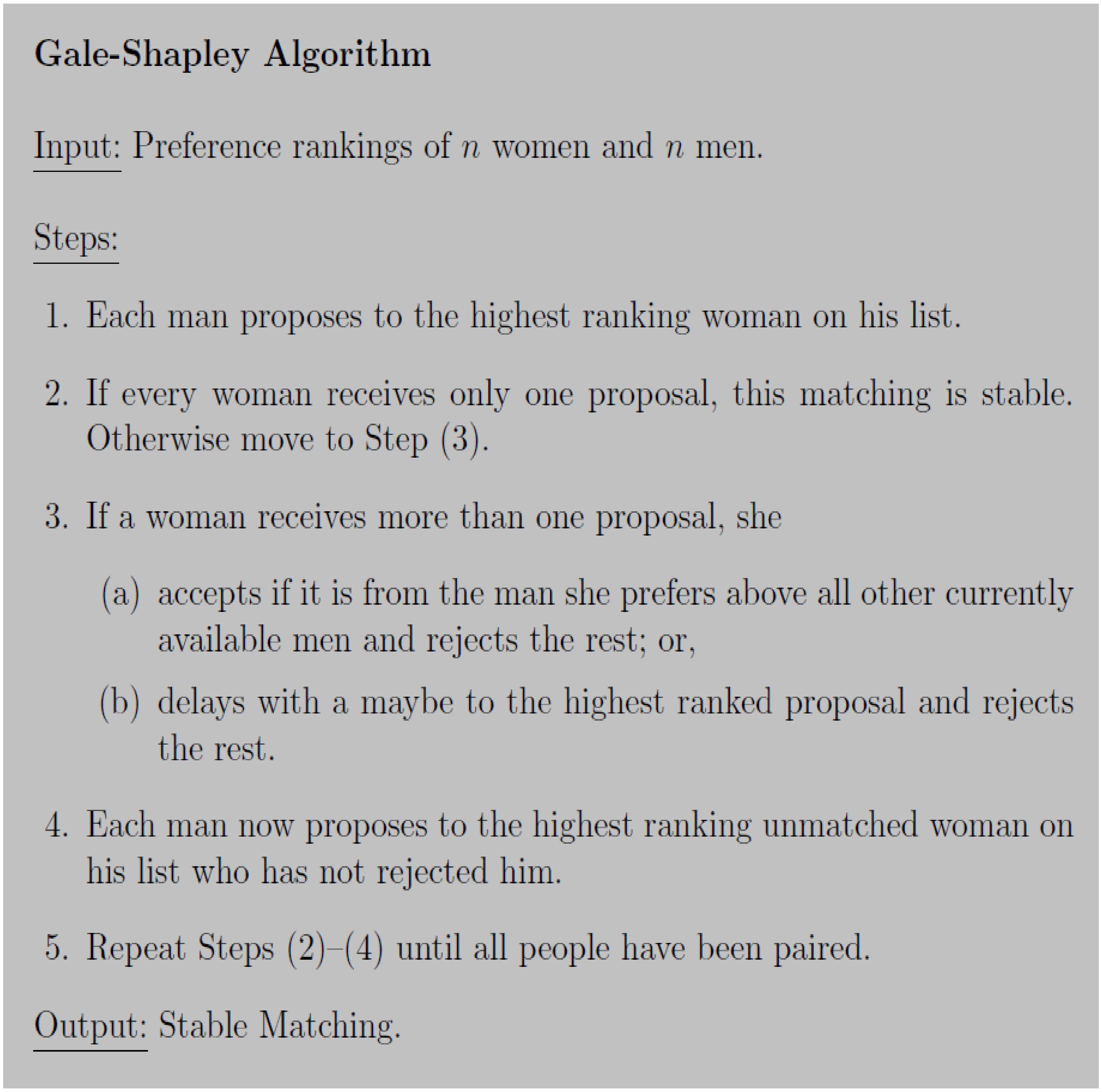


This final matching is in fact stable due to the following: Will and Carol are paired, and each other's first choice; Anne only prefers Ted over Rob, but Ted does not prefer Anne over Diana; Brenda does not prefer anyone over Stan; Diana prefers either Rob or Stan over Ted, but neither of them prefers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

Remark:

While searching for an unstable pair and switching the matching may eventually result in a stable matching (in fact, it will always eventually land on a stable matching), a better procedure exists for finding a stable matching.

The algorithm below is named for **David Gale and Lloyd Shapley**, the two American mathematicians and economists who published this algorithm in 1962.



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.

- The Gale-Shapley Algorithm is written so that the men are always proposing, giving the women the choice to accept, reject, or delay.

- This produces an asymmetric algorithm, meaning a different outcome could occur if the women were proposing, which is demonstrated in the next two examples.