Algebra and Discrete Mathematics
ADM

Bc. Xiaolu Hou, PhD.

FIIT, STU
xiaolu.hou @ stuba.sk

Course Outline

- Vectors and matrices
- System of linear equations
- Matrix inverse and determinants
- Vector spaces and matrix transformations
- Fundamental spaces and decompositions
- Eulerian tours
- Hamiltonian cycles
- Midterm
- Paths and spanning trees
- Trees and networks
- Matching
- Tutorial 12

Recommended reading

- Saoub, K. R. (2017). A tour through graph theory. Chapman and Hall/CRC.
  - Sections 5.1, 5.2, 5.3
  - Free copy online

Lecture outline

- Bipartite graphs
- Matching terminology and strategies
- Augmenting paths and vertex covers
- Stable marriages
- Unacceptable partners

Trees and networks

- Bipartite graphs
- Matching terminology and strategies
- Augmenting paths and vertex covers
- Stable marriages
- Unacceptable partners

Bipartite graphs

Definition

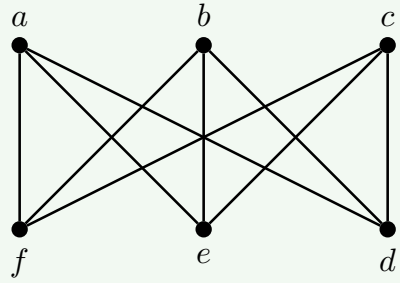
A graph $G = (V, E)$ is *bipartite* if the vertices can be partitioned into two sets, V_1 and V_2, so that

$$V_1 \cap V_2 = \emptyset, \quad V = V_1 \cup V_2,$$

and every edge has exactly one endpoint in V_1 and the other endpoint in V_2.

Example

- $X = \{a, b, c\}$
- $Y = \{d, e, f\}$



Different drawing

- The drawing of a graph can vary
- It may not be easy to see if the graph is bipartite
- But, if you are traveling along a path or cycle in a graph, the vertices will need to alternate between the two parts of the vertex set, such as a vertex from X, then Y, then X, etc
- So if a cycle exists in the graph, it must have even length since otherwise two vertices along the cycle would come from the same part

Determine if a graph is bipartite

Theorem

A graph G is bipartite iff there are no odd cycles in G.

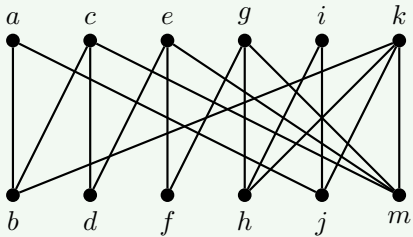
Note

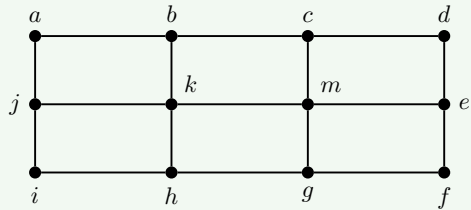
- In practice, we usually search for odd cycles within a graph and if we cannot find any, we try to redraw the graph to emphasize that it is bipartite
- A bipartite graph can have multi-edges (and so need not be simple) but cannot have loops (since these are odd cycles of length 1)

Determine if a graph is bipartite – example

Example

- The graph is bipartite
- $X = \{a, c, e, g, i, k\}$,
  $Y = \{b, d, f, h, j, m\}$

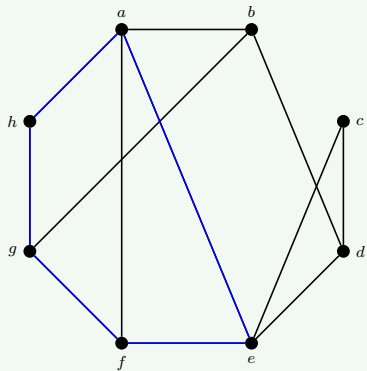


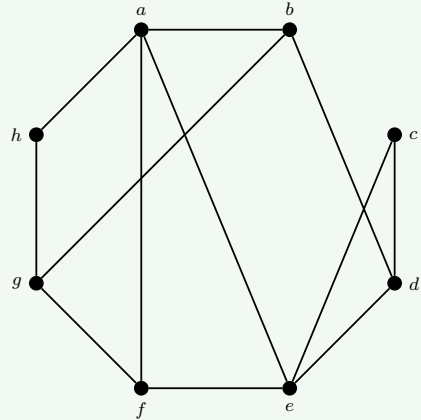


Determine if a graph is bipartite – example

Example

- Not bipartite since there are odd cycles, such as the $5-$cycle $aefgha$





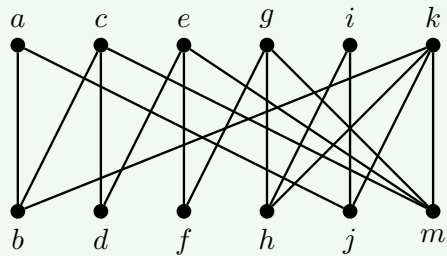
Complete bipartite graph

Definition

A simple bipartite graph $G = (V_1 \cup V_2, E)$ is a *complete bipartite graph* if every vertex in V_1 is adjacent to every vertex in V_2. If $|V_1| = m$, $|V_2| = n$, we write $K_{m,n}$

Example

$K_{6,6}$



- Bipartite graphs
- Matching terminology and strategies
- Augmenting paths and vertex covers
- Stable marriages
- Unacceptable partners

Definitions

Definition

Given a graph $G = (V, E)$

- Two edges that do not share an endpoint are called *independent* edges
- A *matching* M is a subset of independent edges of G
- The size of a matching, denoted $|M|$, is the number of edges in the matching

Applications

- Marriages
- Assign tasks to employees
  - None of the tasks rely on each other but no person has enough time to complete more than one task.
  - In addition, most employees are only qualified to complete some of the tasks

Remark

Matching problems often (though not always) make use of bipartite graphs since the items being matched are usually of two distinct types

Matching – example

Example

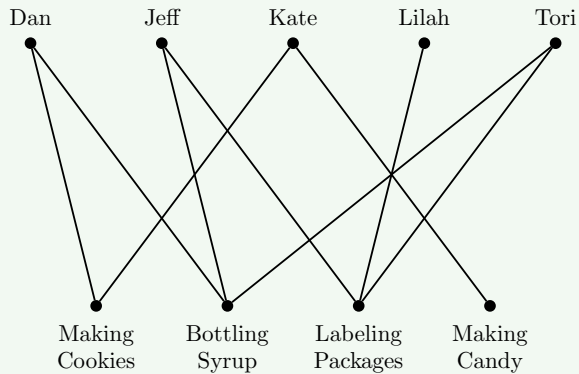
- Each employee is shown along with the jobs for which he or she is qualified
- Model using a bipartite graph where X consists of the employees and Y consists of the tasks

Employee	Task	
Dan	Making Cookies	Bottling Syrup
Jeff	Labeling Packages	Bottling Syrup
Kate	Making Candy	Making Cookies
Lilah	Labeling Packages	
Tori	Labeling Packages	Bottling Syrup

Matching – example

Example

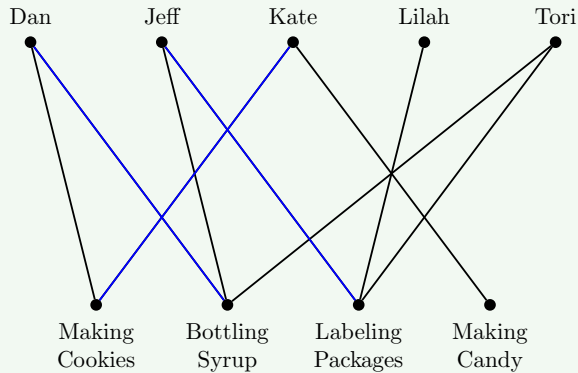
- X consists of the employees and Y consists of the tasks
- We draw an edge between two vertices a and b if employee a is capable of completing the task b



Matching – example

Example

- One possible matching



Matching problem

- What is the important criteria for a solution and how does that translate to a matching
- Is it more important for each employee to have a task or for every task to be completed

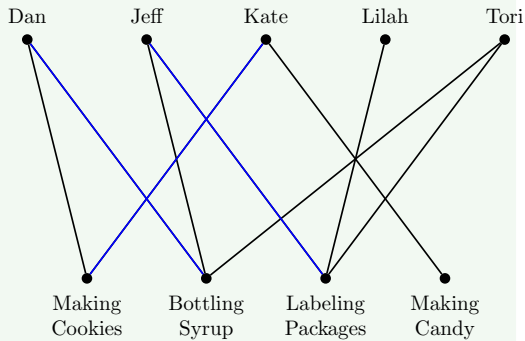
Saturated vertices

Definition

A vertex *saturated* by a matching M if it is incident to an edge of the matching; otherwise, it is *unsaturated*.

Example

- Saturated vertices: Dan, Jeff, Kate, Making Cookies, Bottling Syrup, and Labeling Packages
- Unsaturated vertices: Making candy, Lilah, and Tori



Different kinds of matchings

Definition

Given a matching M on a graph G, M is

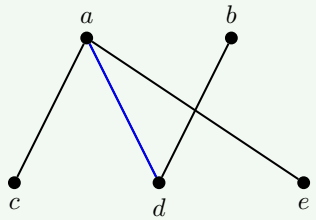
- *maximal* if M cannot be enlarged by adding an edge
- *maximum* if M is of the largest size among all possible matchings
- *perfect* if M saturates every vertex of G
- an $X-$matching if it saturates every vertex from the collection of vertices X

Note

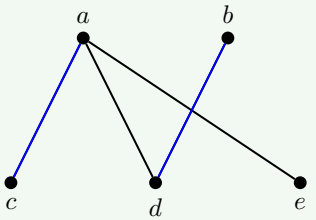
- A perfect matching is automatically maximum and a maximum matching is automatically maximal, though the reverse need not be true
- There can be several matchings of equal size

Maximal and maximum matching – example

Example



Maximal matching



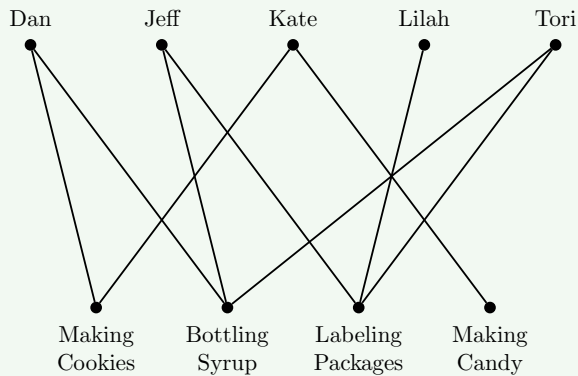
Maximum matching

- Maximal: no other edges in the graph can be added since the remaining edges require the use of a saturated vertex (either a or d)
- Maximum: There is no way for a matching to contain three edges - otherwise a would have two edges incident to it
- Matching on the right is an $X-$matching for $X = \{a, b\}$
- Neither matching is perfect since not every vertex is saturated

Find a matching – example

Example

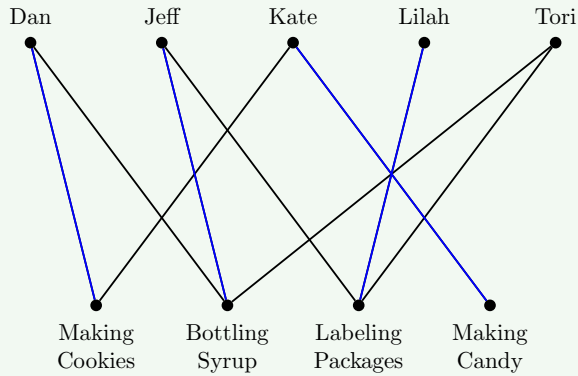
- We need the tasks to be completed but do not need every employee to be assigned a task
- We need an $X-$matching, where X consists of the vertices representing the tasks



Find a matching – example

Example

- One possible matching
- All tasks are assigned to an employee, but not all employees have a task



Hall's Marriage Theorem

Theorem (Hall's Marriage Theorem)

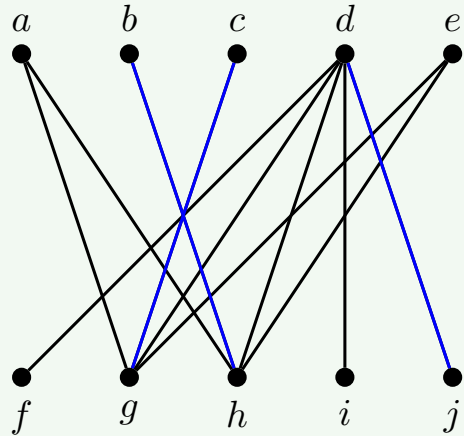
Given a bipartite graph $G = (X \cup Y, E)$, there exists an $X-$matching iff any $S \subseteq X$ satisfies $|S| \leq |N(S)|$

- The neighbor set of a set of vertices S, $N(S)$, consists of all vertices incident to at least one vertex from S
- Hall's Marriage Theorem does not give a definitive answer about the size of a maximum matching but rather gives us the tools to reason why an X-matching does not exist

Hall's Marriage Theorem – example

Example

- Let $X = \{a, b, c, d, e\}$, $S = \{f, i, j\}$
- $N(S) = \{d\}$
- By Hall's Marriage Theorem, there is no $X-$matching.
- Since at most one of the vertices from S can be paired with d, we know the maximum matching can contain at most 3 edges
- The above matching is maximum



Find a maximum matching

- Next, we use a specific type of path and a collection of vertices as a way to determine the answer to the optimization question - how do we find a maximum matching?

- Bipartite graphs
- Matching terminology and strategies
- Augmenting paths and vertex covers
- Stable marriages
- Unacceptable partners

Alternating and augmenting paths

Definition

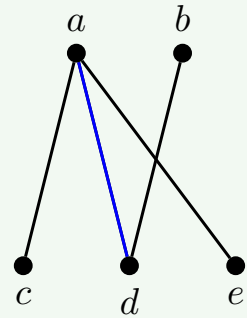
Given a matching M of a graph G, a path is called

- $M-$*alternating* if the edges in the path alternate between edges that are in M and edges that are not in M
- $M-$*augmenting* if it is $M-$alternating and both endpoints of the path are unsaturated by M, implying both the starting and ending edges of the path are not in M

Alternating and augmenting paths – example

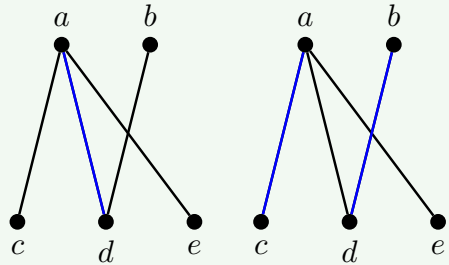
Example

- $cadb$ is alternating and augmenting (both c and b are unsaturated)
- If we switch the edges along this path we get a larger matching
- This switching procedure removes the matched edges and adds the previously unmatched edges along an augmenting path
- Since the path is augmenting, the matching increases in size by one edge



Alternating and augmenting paths – example

Example



- This switching procedure removes the matched edges and adds the previously unmatched edges along an augmenting path
- Since the path is augmenting, the matching increases in size by one edge
- $cadb$ is still alternating, but not augmenting

Berge's Theorem

Theorem (Berge's Theorem)

A matching M of a graph G is maximum iff G does not contain any $M-$augmenting paths.

Note

Unlike Hall's Theorem, Berge's Theorem holds for both bipartite graphs and non-bipartite graphs.

Augmenting Path Algorithm

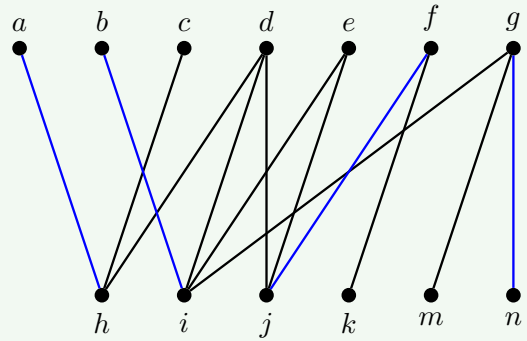
- Input: Bipartite graph $G = (X \cup Y, E)$
- Output: Maximum matching for G

Augmenting Path Algorithm – steps

1. Find an arbitrary matching M
2. $U =$ set of unsaturated vertices in X
3. If $U = \emptyset$, M is a maximum matching; otherwise, select a vertex $x \in U$
4. Consider $y \in N(x)$, if y is unsaturated, go to step 5, otherwise, step 6
5. Add the edge xy to M to obtain a larger matching M' . Return to step 2 and recomputed U.
6. 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'. Return to step 2 and recomputed U.
   b. If the path is not $M-$augmenting, return to step 4, choose a new vertex from $N(x)$
7. Repeat steps 2 - 4 until all vertices from U have been considered

Augmenting Path Algorithm – example

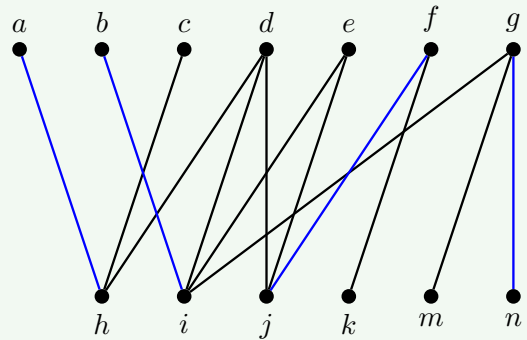
Example



- Step 1. arbitrary matching
- Step 2. $U = \{c, d, e\}$

Augmenting Path Algorithm – example

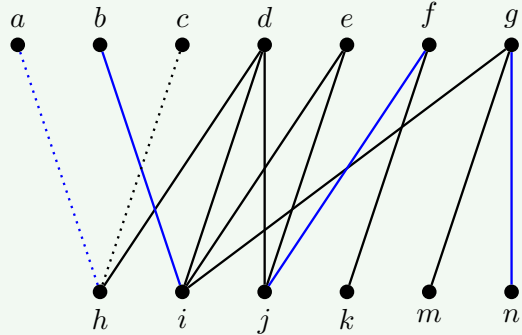
Example



- Step 3. select vertex $c \in U$
- Step 4. only neighbor of c is h, which is saturated
- Step 6. form an $M-$alternating path starting with edge ch, cha

Augmenting Path Algorithm – example

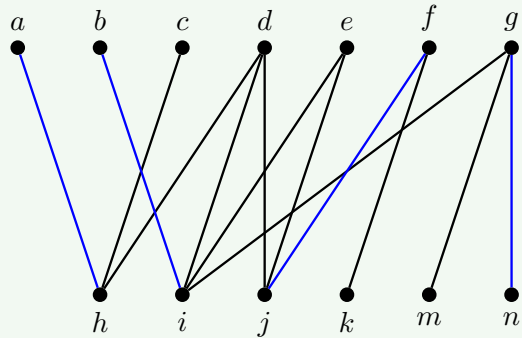
Example



- Step 6. cha is not augmenting, go to step 4
- Step 4. no other neighbor to choose, go to step 3

Augmenting Path Algorithm – example

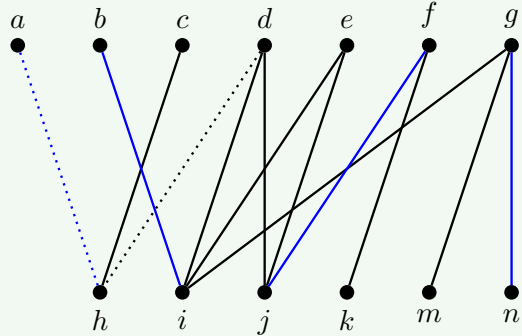
Example



- $U = \{c, d, e\}$
- Step 3. select vertex $d \in U$
- Step 4. $N(d) = \{h, i, j\}$, consider h

Augmenting Path Algorithm – example

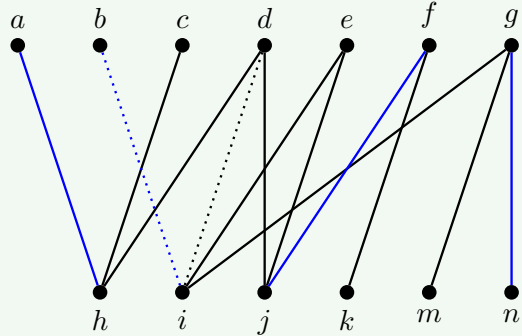
Example



- Step 6. dha is not $M-$augmenting, go to step 4
- Step 4. $N(d) = \{h, i, j\}$, consider i

Augmenting Path Algorithm – example

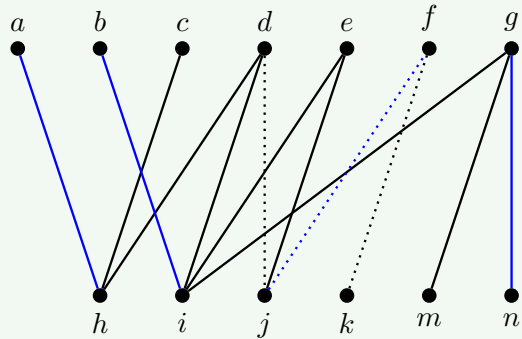
Example



- Step 6. dib is not $M-$augmenting, go to step 4
- Step 4. $N(d) = \{h, i, j\}$, consider j

Augmenting Path Algorithm – example

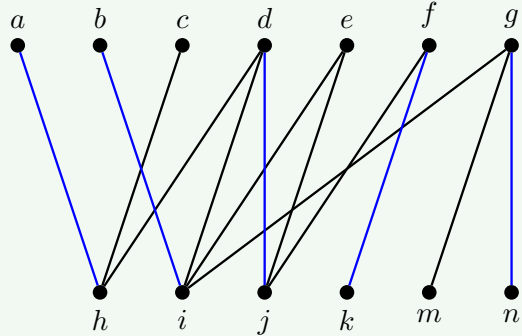
Example



- Step 6. $djfk$ is augmenting, form a new matching: remove fj, add dj, fk

Augmenting Path Algorithm – example

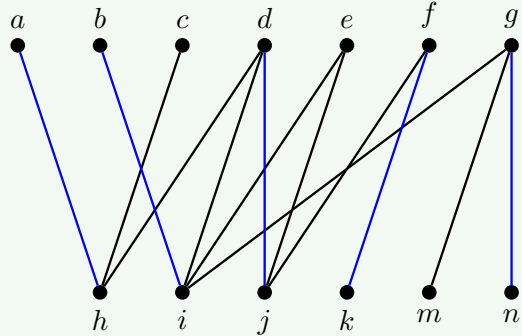
Example



- Step 2. $U = \{c, e\}$
- Step 3. select $c \in U$
- Step 4. only neighbor of c is h, which is saturated, go to step 6
- Step 6. cha is not $M-$augmenting

Augmenting Path Algorithm – example

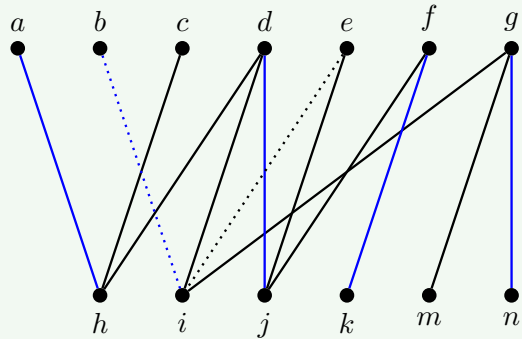
Example



- $U = \{c, e\}$
- Step 3. select $e \in U$
- Step 4. neighbors of e: i, j, consider i
- Step 6. eib

Augmenting Path Algorithm – example

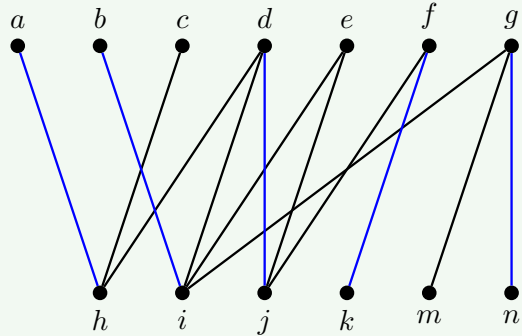
Example



- Step 6. eib is not $M-$augmenting
- Step 4. neighbors of e: i, j, consider j
- Step 6. $ejdha$, $ejdib$

Augmenting Path Algorithm – example

Example



- Step 6. $ejdha$, $ejdib$, neither is $M-$augmenting
- No $M-$augmenting path exists, we have obtained a maximum matching

Vertex cover

Definition

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 G.

- Every graph has a vertex cover, e.g. take Q to be the set of all vertices
- But, we would like to optimize the vertex cover, i.e. find a minimum vertex cover

König-Egnerváry Theorem

Theorem

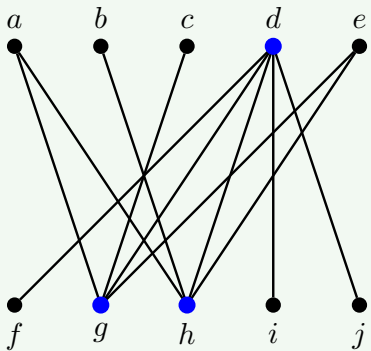
For a bipartite graph G, the size of a maximum matching of G equals the size of a minimum vertex cover for G.

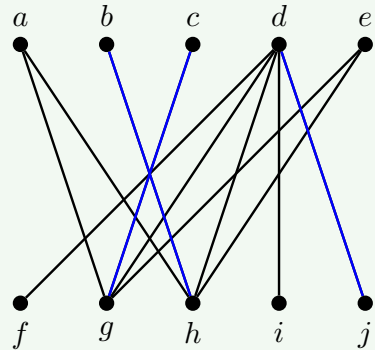
- Given any vertex in the cover, at most one matched edge can be incident to it

König-Egnerváry Theorem

Example

- We have discussed that the matching is maximum
- One vertex cover with 3 vertices is $\{g, h, d\}$





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 iteration of steps 2-6
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

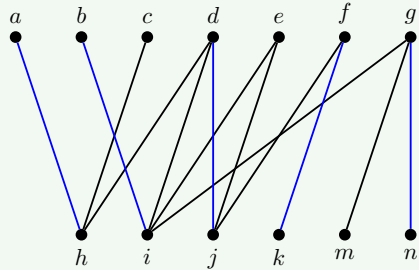
Note

- Step 2 includes not just the vertices in U from Augmenting Path Algorithm, but also any vertex that was reached through an alternating path that originated at a vertex from U.
- Thus the unmarked vertices will be those that are never mentioned during the final step of the implementation of the Augmenting Path Algorithm

Augmenting Path Algorithm final iteration – example

Example

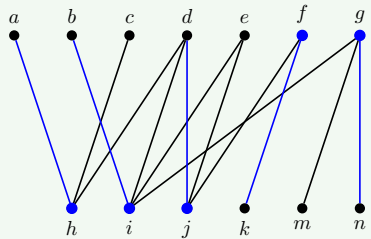
- Step 2. $U = \{c, e\}$
- Step 3. select $c \in U$
- Step 4. only neighbor of c is h, which is saturated, go to step 6
- Step 6. cha is not $M-$augmenting
- Step 3. select $e \in U$
- Step 4. neighbors of e: i, j, consider i
- Step 6. eib is not $M-$augmenting
- Step 4. neighbors of e: i, j, consider j
- Step 6. $ejdha$, $ejdib$ are not $M-$augmenting
- Mark vertices: c, e, h, a, i, j, b, d

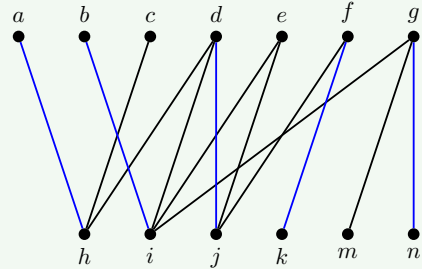


Vertex Cover Method – example

Example

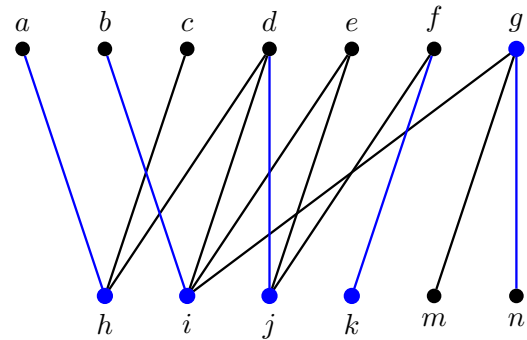
- Mark vertices: c, e, h, a, i, j, b, d
- Unmarked vertices in X: f, g
- Marked vertices in Y:h, i, j
- Size of vertex cover: 5





Remark

- More than one minimum vertex cover may exist for the same graph, just as more than one maximum matching may exist
- In the previous example, we found one such vertex cover through the matching found using the Augmenting Path Algorithm
- Another minimum vertex cover



Remark

- Given a maximum matching, apply Augmenting Path Algorithm starting from step 2 gives a minimum vertex cover
- If we a $X-$matching, treat X as Y and Y as X in the vertex cover method
- If perfect matching, then either X or Y gives a minimum vertex cover because size of minimum vertex cover is equal to size of maximum matching
- We will see more examples in Tutorial 12

Trees and networks

- Bipartite graphs
- Matching terminology and strategies
- Augmenting paths and vertex covers
- Stable marriages
- Unacceptable partners

Stable Marriage Problem

- Our terminology will reflect the marriage model
- We start with two distinct, yet equal sized, groups of people, usually men and women, who have ranked the members of the other group
- A matching is considered *stable* if there are no two people who would both prefer each other over their current partners. That is, there are no mutually preferred alternatives that would rather "run away together."

Stable matching

Definition

Let $K_{n,n} = (X \cup Y, E)$ be a complete bipartite graph, where each vertex has a strict preference ordering over its neighbors. Let M be a perfect matching in G.

- A pair of unmatched vertices $(x, y) \in X \times Y$ is *unstable* if
  - x prefers y over their current partner, and
  - y prefers x over their current partner
- M is *stable* if no unmatched pair of vertices is unstable

Unstable matching – example

Example

- Four men and four women are being paired into marriages
- Each person has ranked the members of the opposite sex

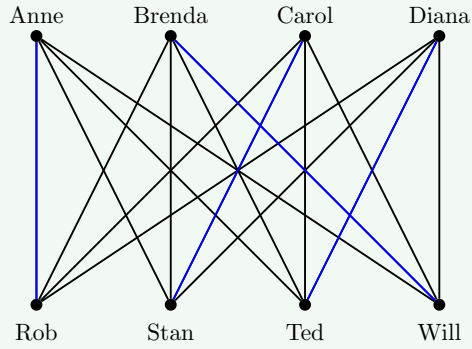
Anne:	$t > r > s > w$	Rob:	$a > b > c > d$
Brenda:	$s > w > r > t$	Stan:	$a > c > b > d$
Carol:	$w > r > s > t$	Ted:	$c > d > a > b$
Diana:	$r > s > t > w$	Will:	$c > b > a > d$

Unstable matching – example

Example

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:	$c > b > a > d$

- Unstable: Will and Carol prefer each other to their current mate



Gale-Shapley Algorithm

- Named after David Gale and Lloyd Shapley, the two American mathematicians and economists who published this algorithm
- In addition, their work led to further studies on economic markets, one of which awarded Shapley (along with his collaborator Alvin Roth) the 2012 Nobel Prize in Economics
- Input: preference rankings of n women and n men
- Output: stable matching

Gale-Shapley Algorithm – 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. Each woman
   i. accepts a proposal if it is from the man she prefers above all other currently available men and rejects the rest (if any); or
   ii. delays with a maybe to the highest ranked proposal and rejects the rest (if any)
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

Gale-Shapley Algorithm – example

Example

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:	$c > b > a > d$

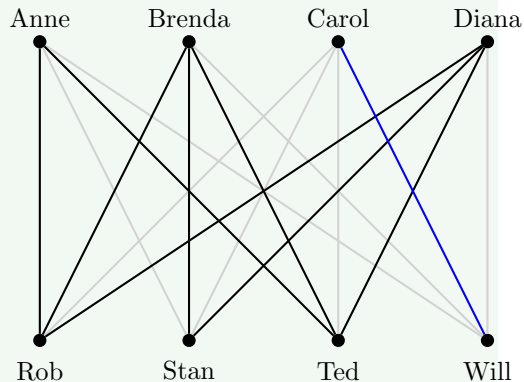
- Step 1. Proposals: Rob-Anne, Stan-Anne, Ted-Carol, Will-Carol
- Step 3. Anne and Carol makes choices
  - Neither Rob nor Stan are Anne's top choice so she rejects the lower ranked one (Stan) and says maybe to the other (Rob)
  - Will is Carol's top choice so she accepts his proposal and rejects Ted

Gale-Shapley Algorithm – example

Example

Rob:	$a > b > c > d$
Stan:	$a > c > b > d$
Ted:	$c > d > a > b$
Will:	$c > b > a > d$

- Step 4. unmatched men propose to the next available woman on their preference list
  - Rob proposes again to Anne since she delayed in the last round
  - Stan proposes to Brenda; cannot propose to Anne since she rejected him previously
  - Ted proposes to Diana



Gale-Shapley Algorithm – example

Example

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:	$c > b > a > d$

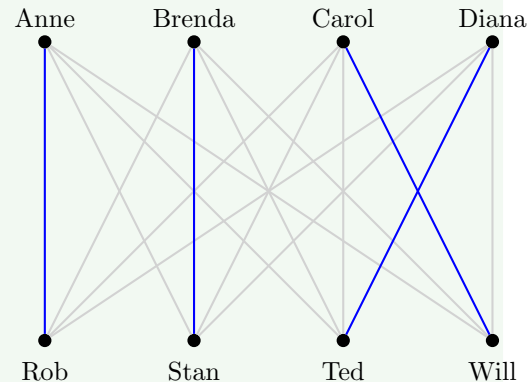
- Step 4. Proposals: Rob-Anne, Stan-Brenda, Ted-Diana
- Step 2. Since all the proposals are different (no woman received more than one proposal), the women must all accept

Gale-Shapley Algorithm – example

Example

Anne:	$t > r > s > w$
Brenda:	$s > w > r > t$
Carol:	$w > r > s > t$
Diana:	$r > s > t > w$

Rob:	$a > b > c > d$
Stan:	$a > c > b > d$
Ted:	$c > d > a > b$
Will:	$c > b > a > d$



- Two men are paired with their top choice, one with his second, and one with his third
- The same holds for the women (two first choices, one second, and one third)

Gale-Shapley Algorithm

- The Gale-Shapley Algorithm is asymmetric and in fact favors the group making the proposals
- In the example above
  - Two men are paired with their top choice, one with his second, and one with his third
  - The same holds for the women (two first choices, one second, and one third)
- If the women were the ones proposing we would likely see an improvement in their overall happiness

Gale-Shapley Algorithm – example

Example

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:	$c > b > a > d$

- Step 1. Proposals: Anne-Ted, Brenda-Stan, Carol-Will, Diana-Rob
- Step 2. Since all the proposals are different (no man received more than one proposal), the men must all accept
- The women were all paired with their first choice, whereas only one man was paired with his first choice, two with their third choice, and one with his fourth choice

Remarks

- The group proposing in the Gale-Shapley Algorithm is more likely to be happy
- In the previous example, even though the second matching seem more imbalanced than the first one, it is still a stable matching – there is no guarantee that a unique stable matching exists
- In fact, many examples have more than one stable matching possible.
- The important concept to remember is that for a complete bipartite graph with rankings, a stable matching will always exist.
- If we generalize this to other types of graphs, the same may not hold.

Trees and networks

- Bipartite graphs
- Matching terminology and strategies
- Augmenting paths and vertex covers
- Stable marriages
- Unacceptable partners

Unacceptable partner

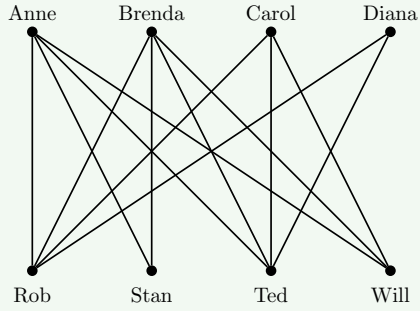
- By having each person rank all others of the opposite sex, we assume that all of these potential matches are acceptable
- This is very clearly not accurate to a real world scenario – some people should never be married if even they are the only pair left
- To adjust for this, we introduce the notion of an unacceptable partner

Unacceptable partner – example

Example

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

- If a person is missing from the ranking, then they are deemed unacceptable
- Will is unacceptable to Diana and only Anne and Brenda are acceptable to Stan
- We consider men proposing, so leave out edges to unacceptable partners for men only



Gale-Shapley Algorithm (with Unacceptable Partners)

- We are still looking for a matching in a bipartite graph, only now the graph is not complete
- We must adjust our notion of a stable matching, since it is possible that not all people could be matched
- Under these new conditions, a matching (with unacceptable partners) is *stable* if no unmatched pair x and y such that x and y are both acceptable to each other, and each is either single or prefers the other to their current partner
- Input: preference ranking of n women and n men
- Output: stable matching

Gale-Shapley Algorithm (with Unacceptable Partners) – steps

1. Each man proposes to the highest ranking woman on his list
2. If every woman receives only one proposal from someone they deem acceptable, they all accept and this matching is stable. Otherwise move to step 3
3. Each woman
   i. rejects a proposal if it is from an unacceptable man;
   ii. accepts if the proposal is from the man she prefers above all other currently available men and rejects the rest (if any); or
   iii. delays with a maybe to the highest ranked proposal and rejects the rest (if any)
4. Each man now proposes to the highest ranking unmatched woman on their list who has not rejected him
5. Repeat steps 2-4 until all people have been paired or until no unmatched man has any acceptable partners remaining

Remark

- Always produces a stable matching
- Can be modified so that the women are proposing

Gale-Shapley Algorithm (with Unacceptable Partners) – example

Example

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

- Step 1. The initial proposals are Rob –Anne, Stan –Anne, Ted –Carol, and Will – Carol.
- Step 3. Anne and Carol makes choices
  - Stan is unacceptable to Anne, she rejects him and since Rob is not her top choice she says maybe
  - Will is Carol's top choice so she accepts his proposal and rejects Ted

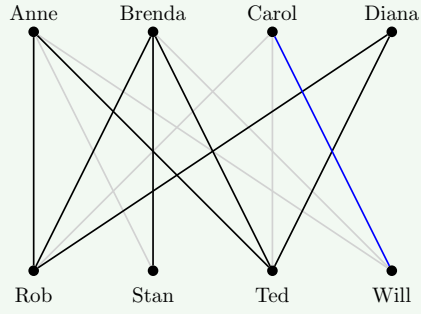
Gale-Shapley Algorithm (with Unacceptable Partners) – example

Example

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

Step 4.

- Remaining men propose to the next available woman on their preference list
- Rob - Anne
- Stan - Brenda
- Ted - Diana



Gale-Shapley Algorithm (with Unacceptable Partners) – example

Example

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

- Step 4. Rob - Anne, Stan - Brenda, Ted - Diana
- Step 3. Even though all proposals are different, Brenda rejects Stan since he is an unacceptable partner. The other two women say maybe since their proposals are not from their top choice of the available men

Gale-Shapley Algorithm (with Unacceptable Partners) – example

Example

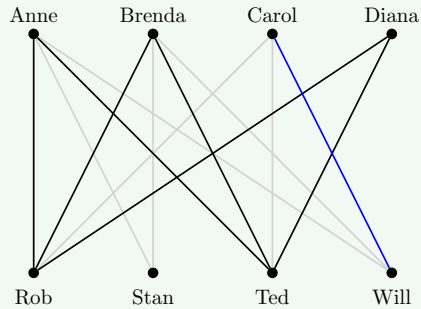
Anne:	$t > r > w$	Rob:	$a > b > c > d$
Brenda:	$w > r > t$	Stan:	$a > b$
Carol:	$w > r > s > t$	Ted:	$c > d > a > b$
Diana:	$s > r > t$	Will:	$c > b > a$

Step 4.

- Rob - Anne
- Ted - Diana
- Stan does not have any acceptable partners left so must remain single

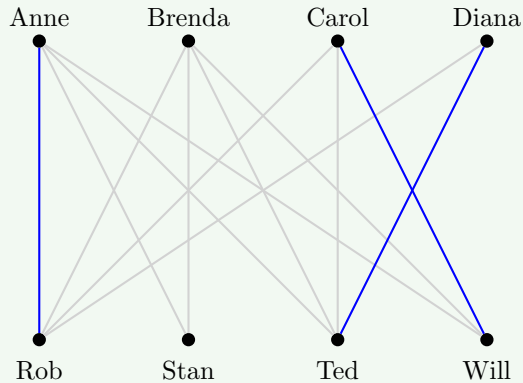
Step 2.

- Both proposals are different and from acceptable partners
- Both women accept



Gale-Shapley Algorithm (with Unacceptable Partners) – example

Example



In a scenario with unacceptable partners a stable matching can exist with not all people paired