Maximum Matchings and How to Find Them

Daniel W. Cranston
dcransto@gmail.com

Department of Mathematics
William & Mary
25 February 2026

Background

Problem: You're home on spring break and want to meet up with some old friends.

Background

Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others.

Background

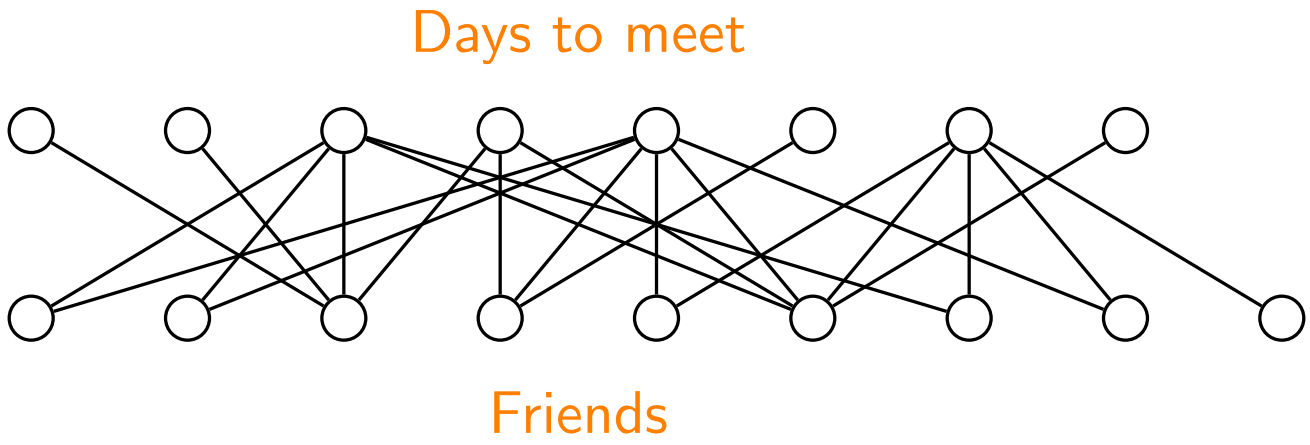
Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible,

Background

Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.

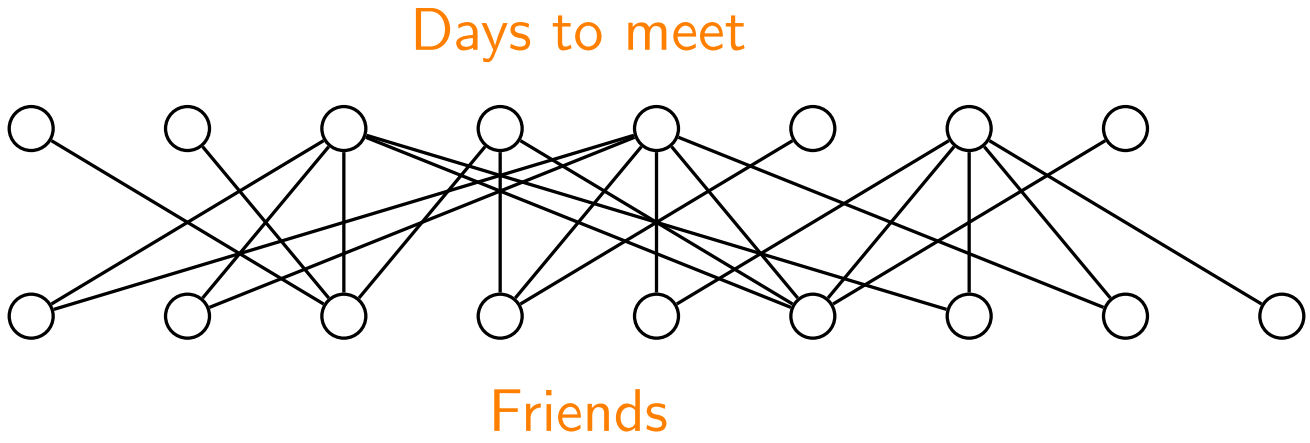
Background

Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Background

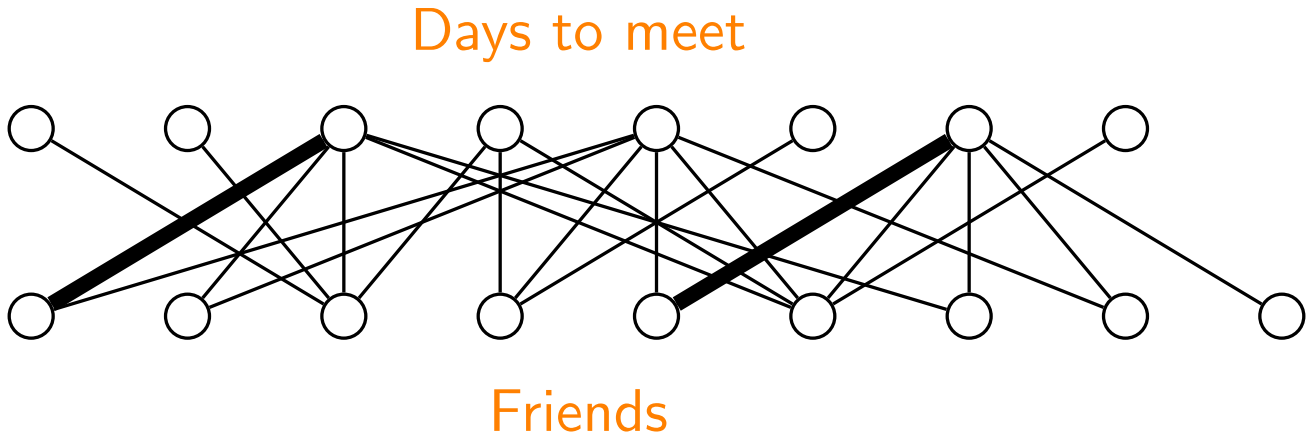
Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Definitons: A matching is an edge set with at most one edge at each vertex.

Background

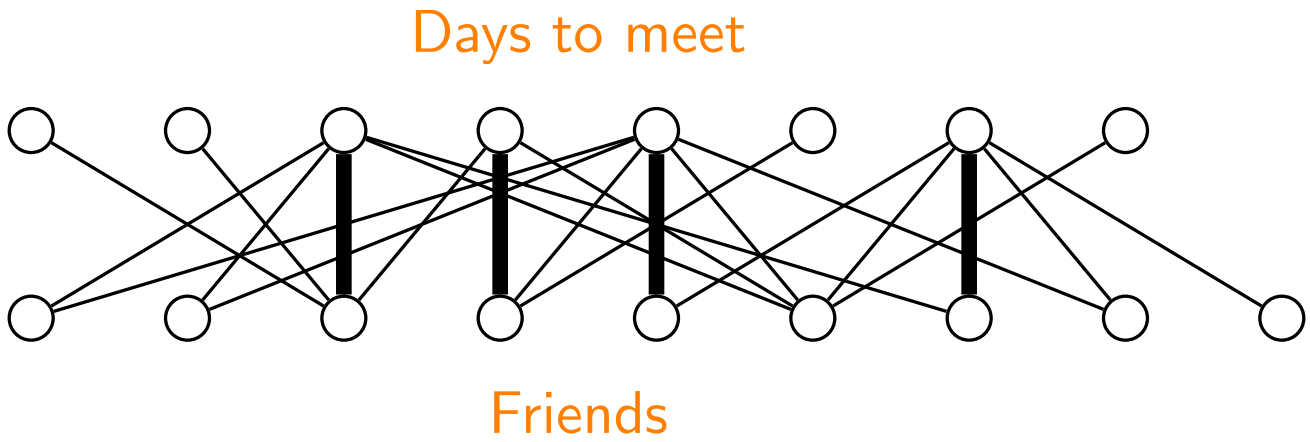
Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Definitons: A matching is an edge set with at most one edge at each vertex.

Background

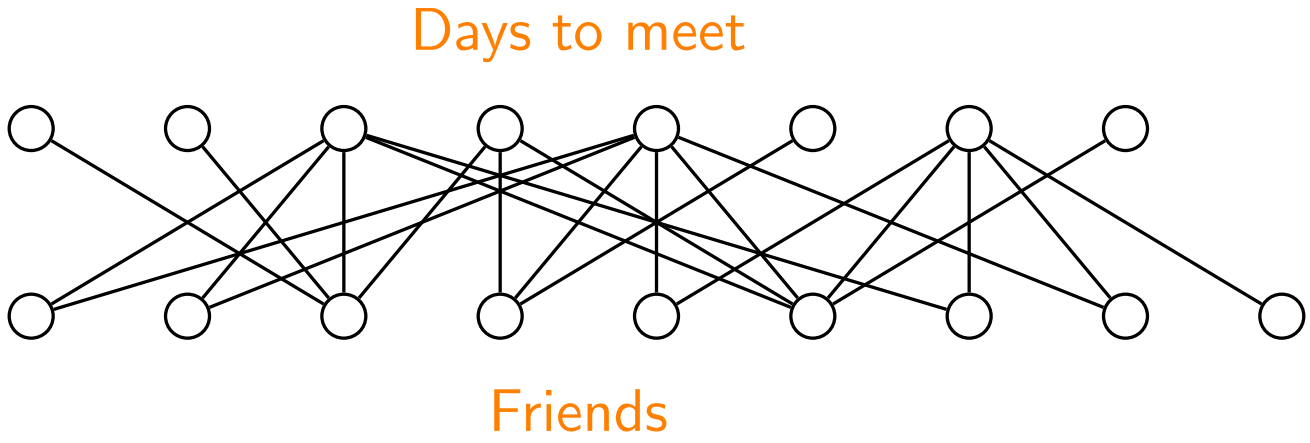
Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Definitons: A matching is an edge set with at most one edge at each vertex.

Background

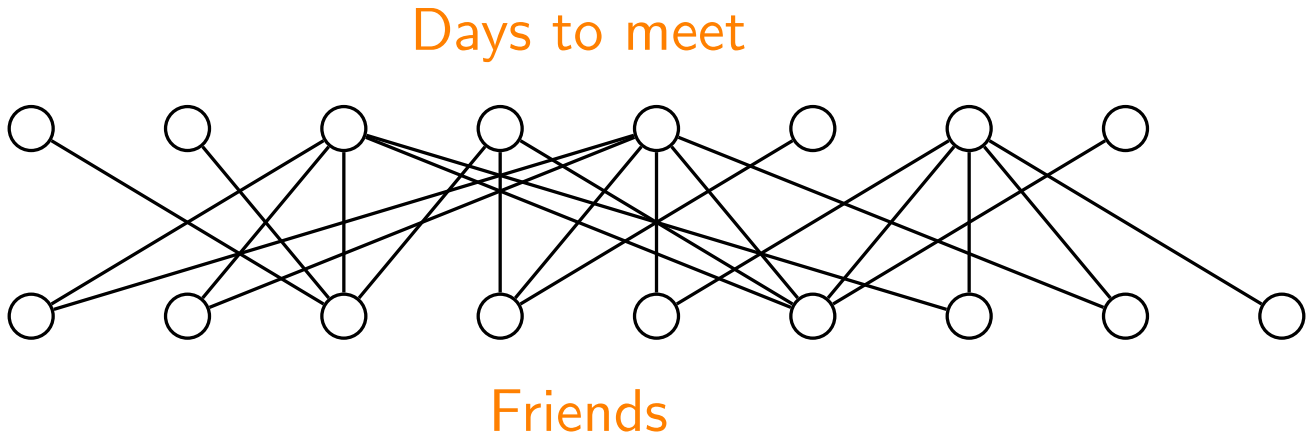
Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Definitons: A matching is an edge set with at most one edge at each vertex. This is the bipartite matching problem.

Background

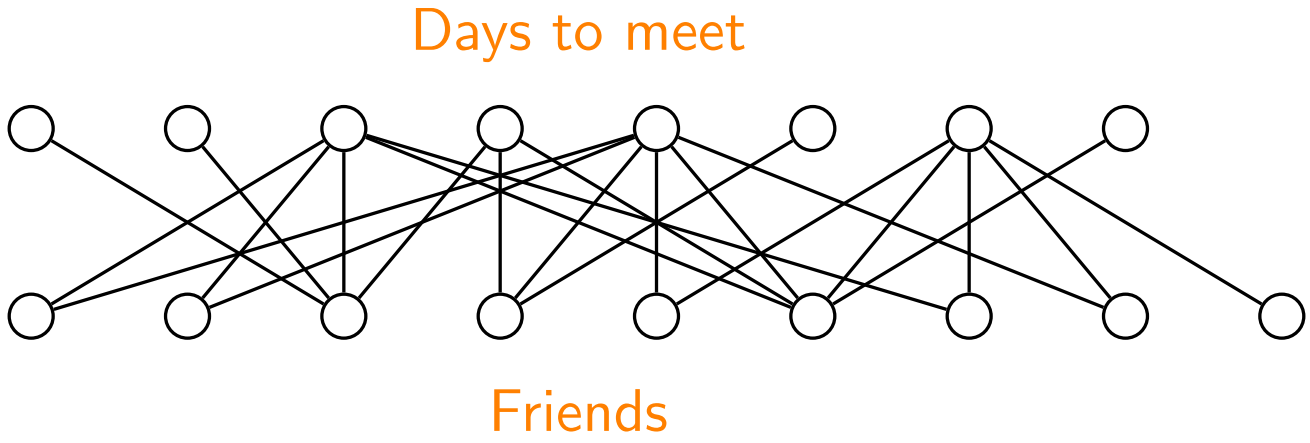
Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Definitons: A matching is an edge set with at most one edge at each vertex. This is the maximum bipartite matching problem.

Background

Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



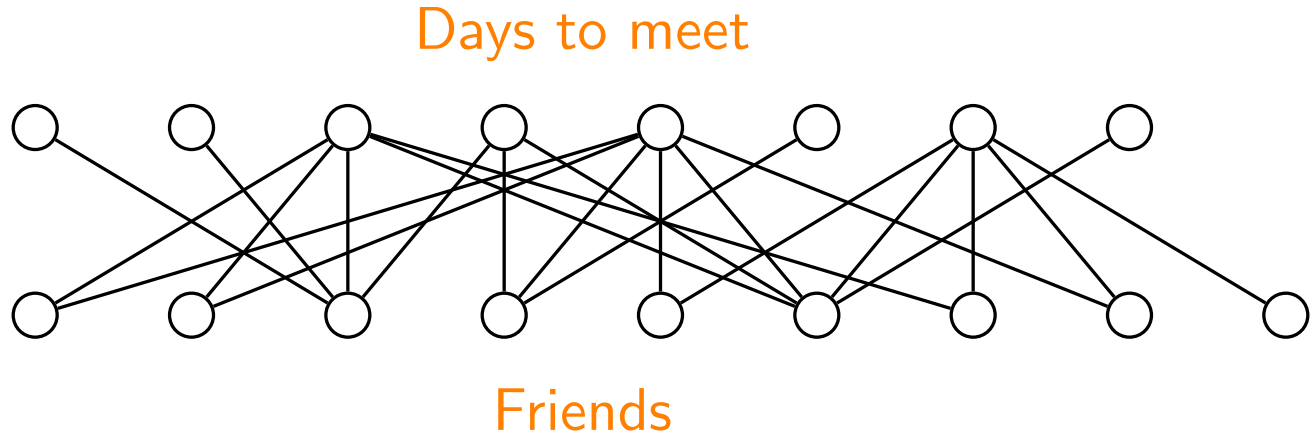
Definitons: A matching is an edge set with at most one edge at each vertex. This is the maximum bipartite matching problem.

Questions:

1. How can you tell if a matching M is maximum?

Background

Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



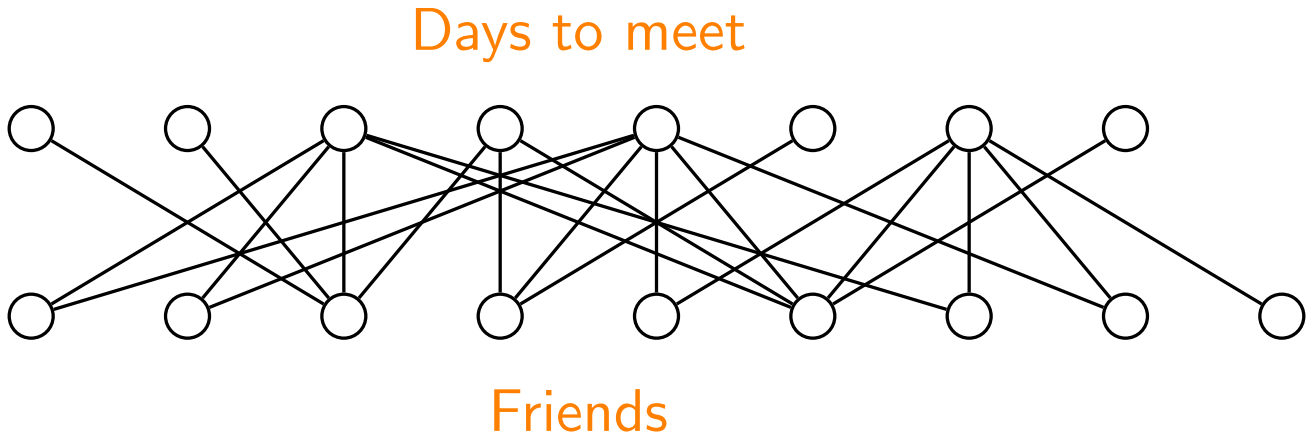
Definitons: A matching is an edge set with at most one edge at each vertex. This is the maximum bipartite matching problem.

Questions:

1. How can you tell if a matching M is maximum?
2. If M is not maximum, can you find a way to improve it?

Background

Problem: You're home on spring break and want to meet up with some old friends. Each friend is available at certain times, but not others. Your goal is to meet as many friends as possible, but you really want time to catch up, so you plan to meet each one-on-one.



Definitons: A matching is an edge set with at most one edge at each vertex. This is the maximum bipartite matching problem.

Questions:

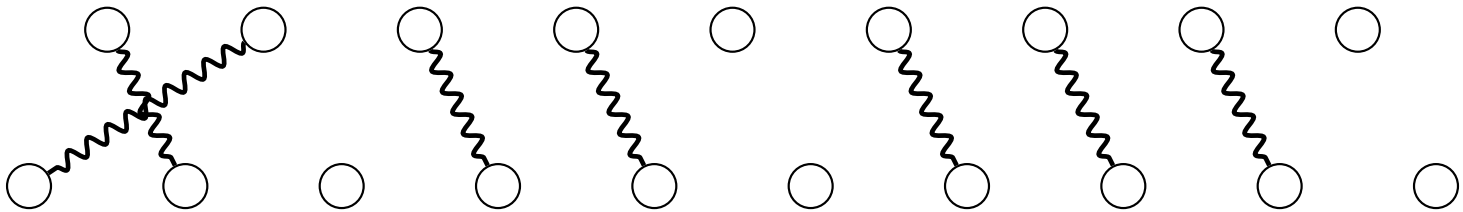
1. How can you tell if a matching M is maximum?
2. If M is not maximum, can you find a way to improve it?
3. How can you find a maximum matching?

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J.

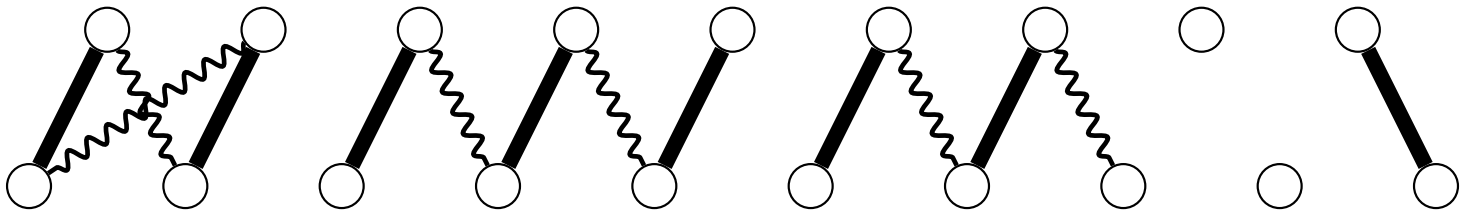
Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J.



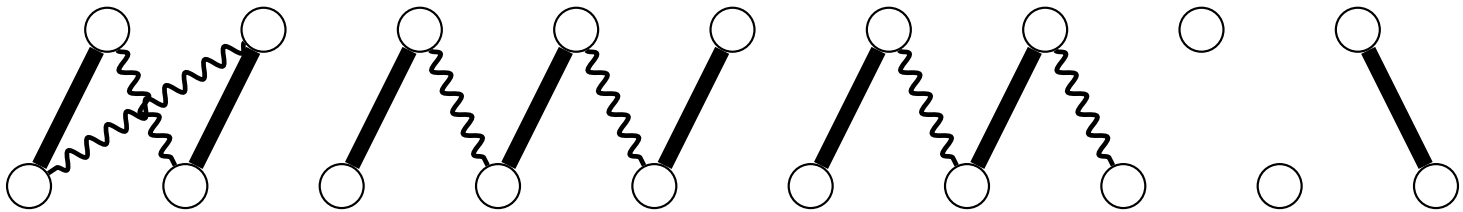
Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J.



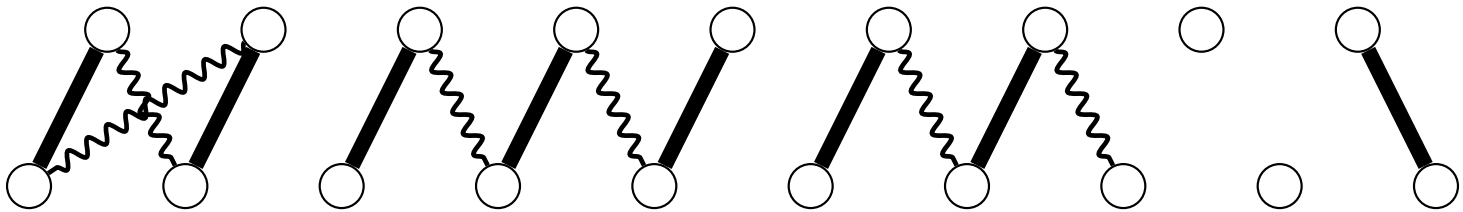
Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M.



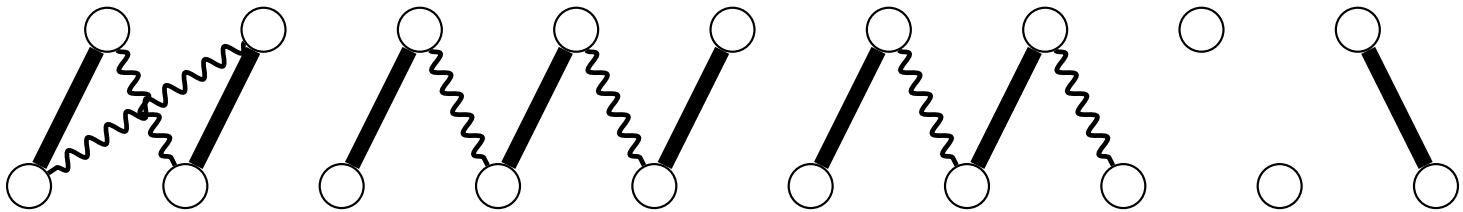
Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$.



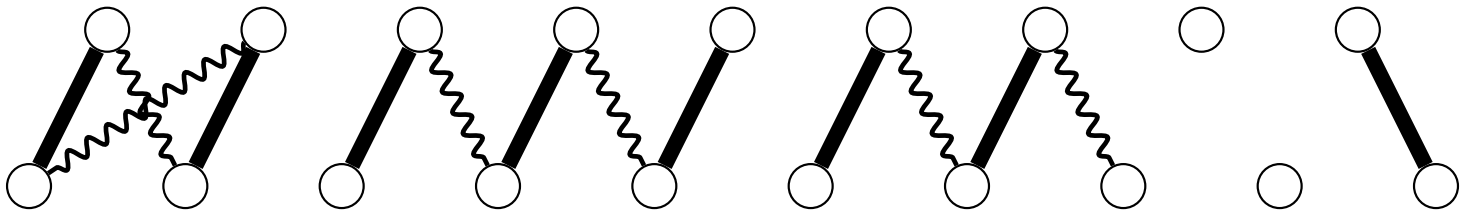
Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.



Maximum Matchings: A Handy Lemma

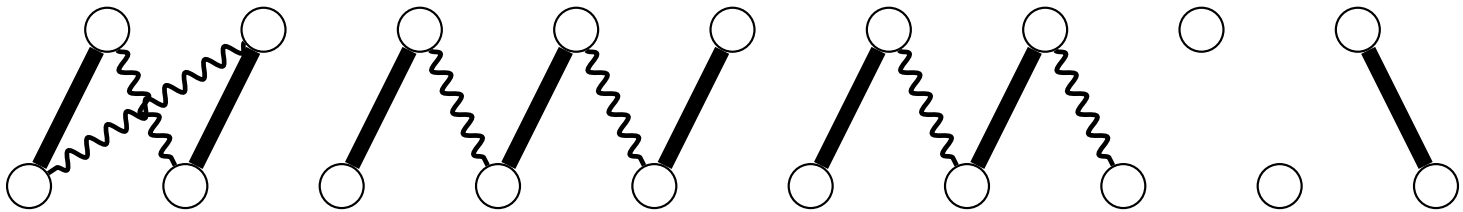
Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.



Lemma: Matching M is maximum iff G has no M-augmenting path.

Maximum Matchings: A Handy Lemma

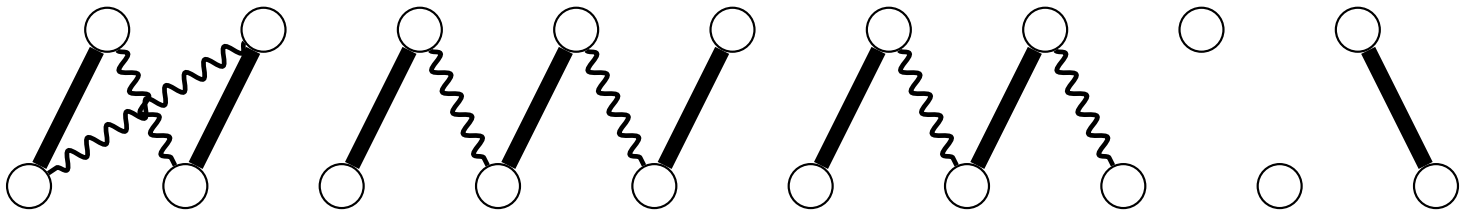
Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.



Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum.

Maximum Matchings: A Handy Lemma

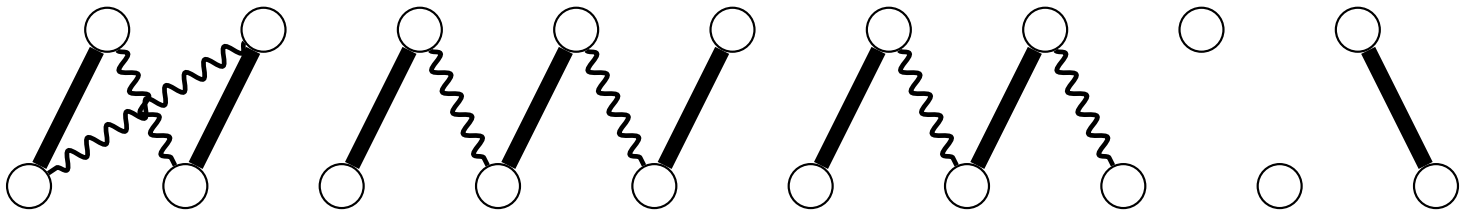
Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.



Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G.

Maximum Matchings: A Handy Lemma

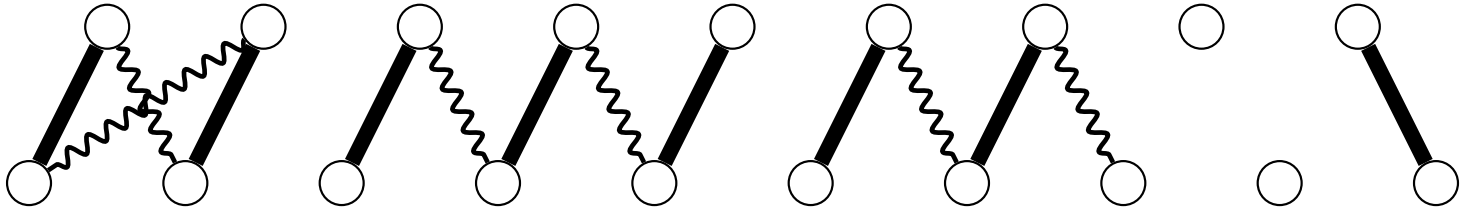
Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.



Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.

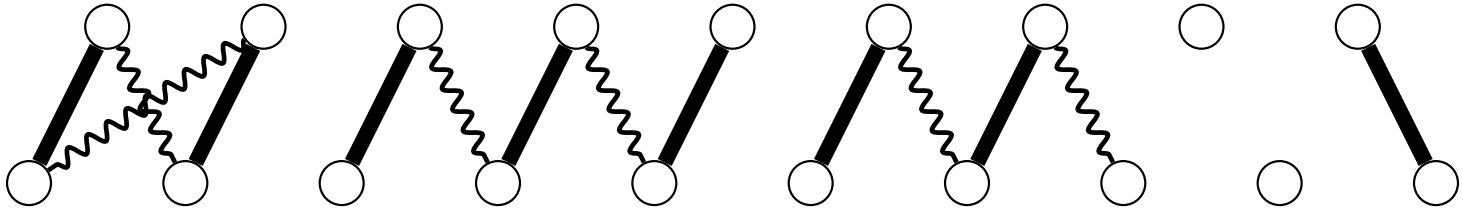


Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Now H has max. degree at most 2, so H is disjoint union of paths and even cycles.

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.

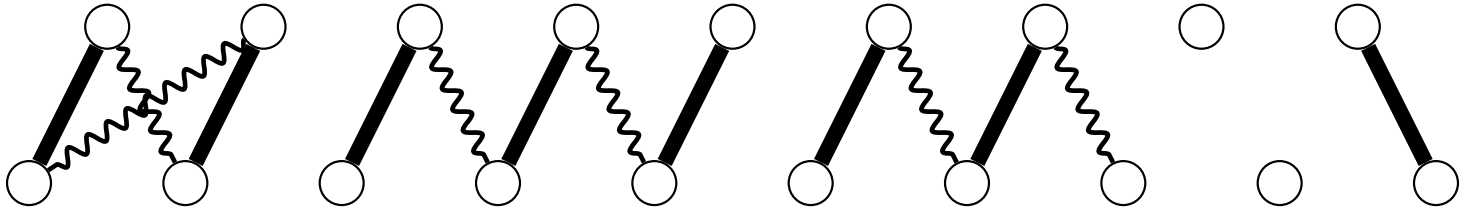


Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Now H has max. degree at most 2, so H is disjoint union of paths and even cycles. Now $|C \cap M| = |C \cap M'|$ for each cycle C.

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.

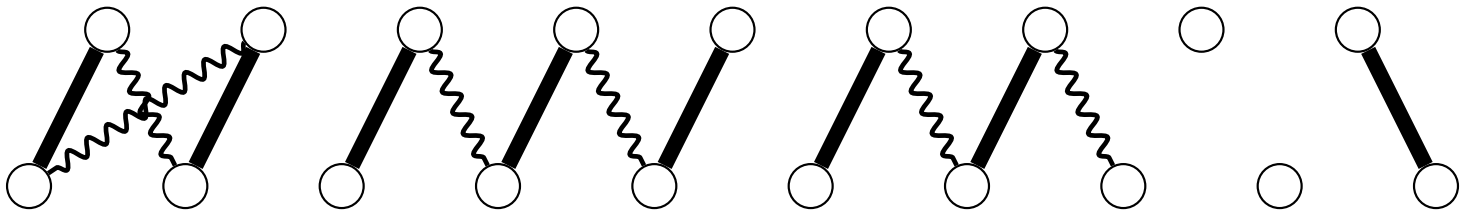


Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Now H has max. degree at most 2, so H is disjoint union of paths and even cycles. Now $|C \cap M| = |C \cap M'|$ for each cycle C. And $||P \cap M'| - |P \cap M|| \leqslant 1$ for each path P.

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.

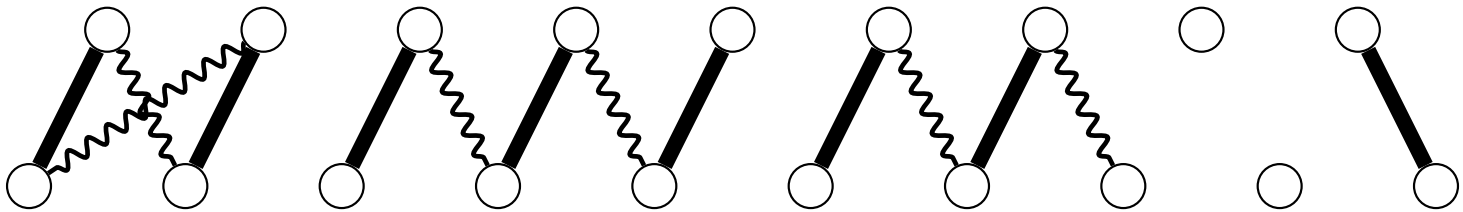


Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Now H has max. degree at most 2, so H is disjoint union of paths and even cycles. Now $|C \cap M| = |C \cap M'|$ for each cycle C. And $||P \cap M'| - |P \cap M|| \leqslant 1$ for each path P. But since $|M'| > |M|$, a path P' has $|M' \cap P'| = |M \cap P'| + 1$.

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.

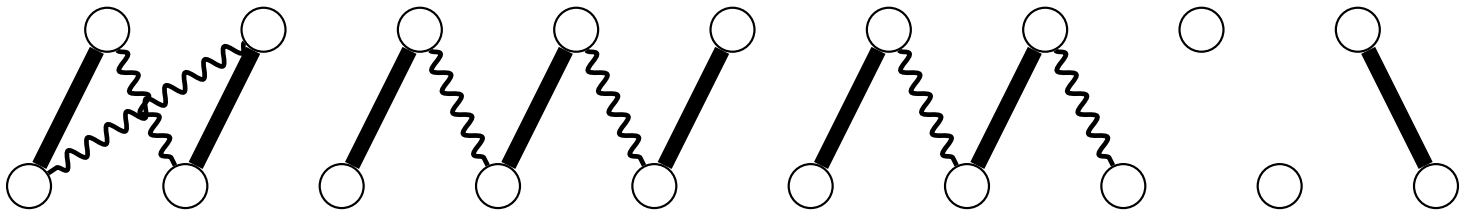


Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Now H has max. degree at most 2, so H is disjoint union of paths and even cycles. Now $|C \cap M| = |C \cap M'|$ for each cycle C. And $||P \cap M'| - |P \cap M|| \leqslant 1$ for each path P. But since $|M'| > |M|$, a path P' has $|M' \cap P'| = |M \cap P'| + 1$. So P' is M-augmenting.

Maximum Matchings: A Handy Lemma

Definitions: The symmetric difference $H \triangle J$ of edge sets H and J is the edges in exactly one of H and J. For G and a matching M in G, an M-augmenting path is path P alternating between edges in and out of M, starting and ending out of M. To augment M by P replace M by $M \triangle P$. If P is M-augmenting, $|M \triangle P| = |M| + 1$.

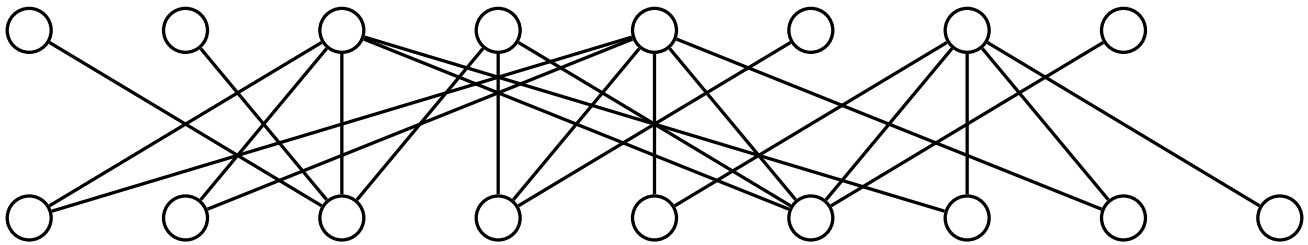


Lemma: Matching M is maximum iff G has no M-augmenting path.
Proof: If G has M-augmenting path P, then $|M \triangle P| = |M| + 1$, so M is not maximum. Now suppose M is not maximum for G. Let M' be a maximum matching, and let $H = M \triangle M'$.

Now H has max. degree at most 2, so H is disjoint union of paths and even cycles. Now $|C \cap M| = |C \cap M'|$ for each cycle C. And $||P \cap M'| - |P \cap M|| \leqslant 1$ for each path P. But since $|M'| > |M|$, a path P' has $|M' \cap P'| = |M \cap P'| + 1$. So P' is M-augmenting. ■

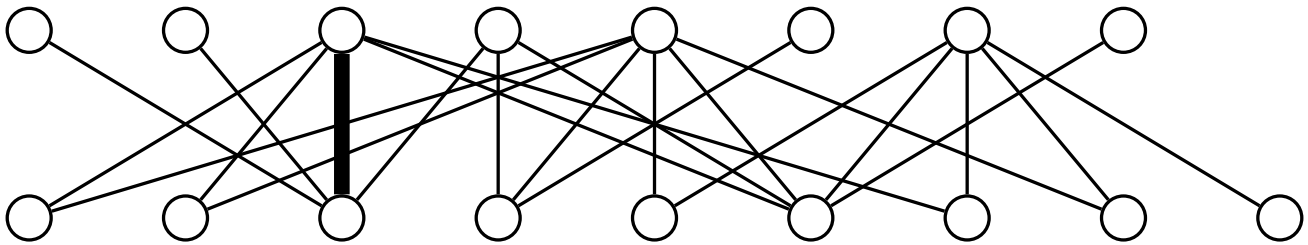
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



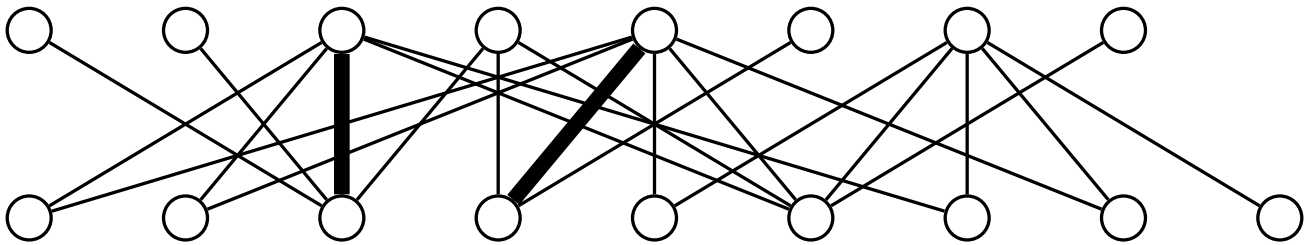
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



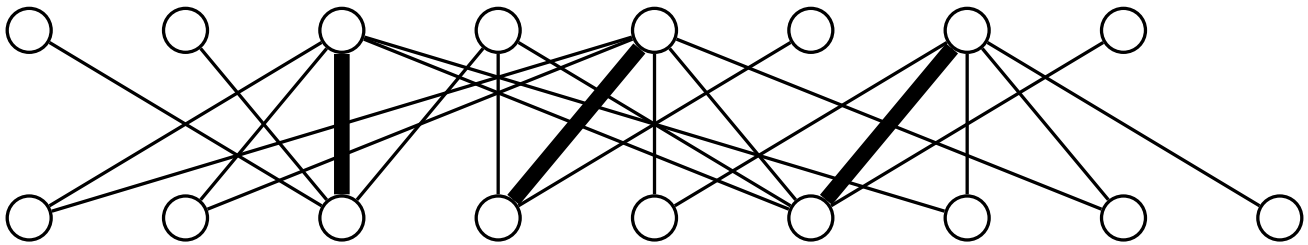
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



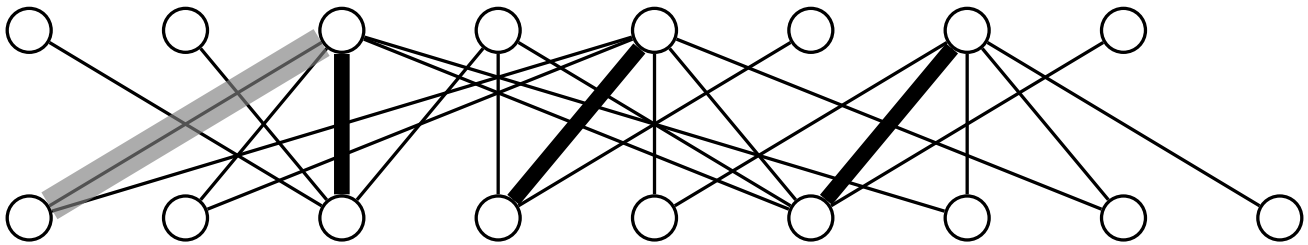
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



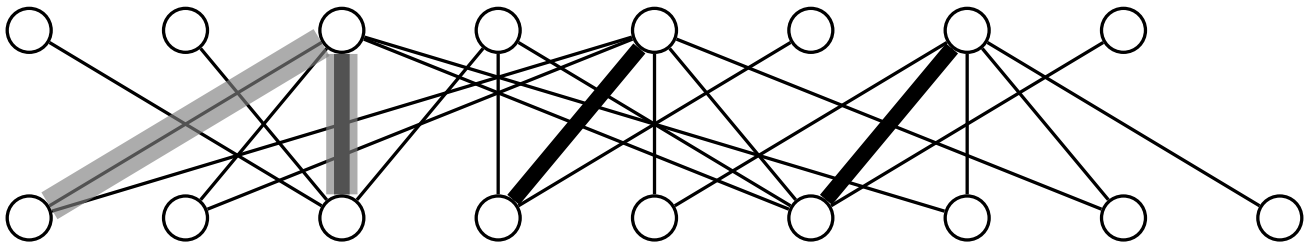
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



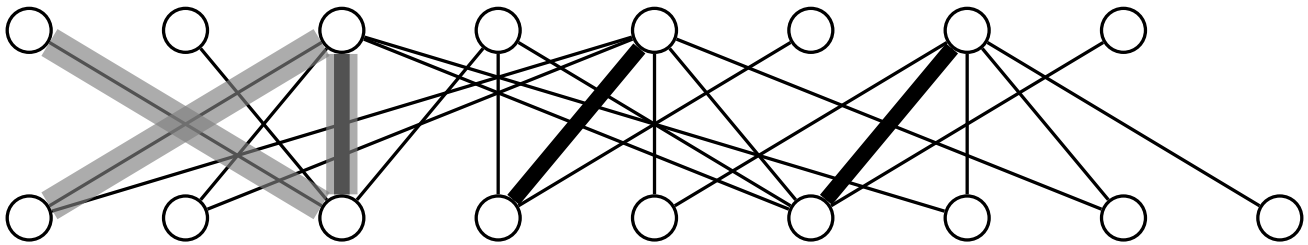
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



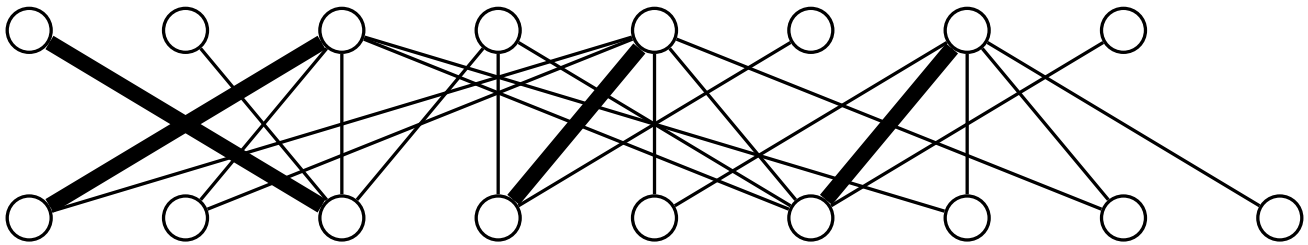
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



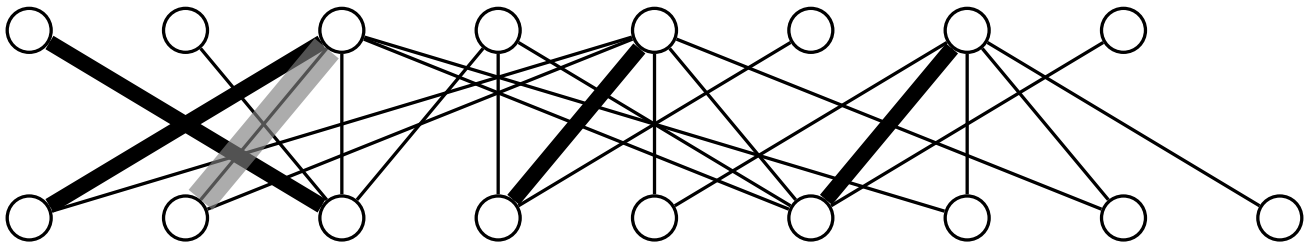
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



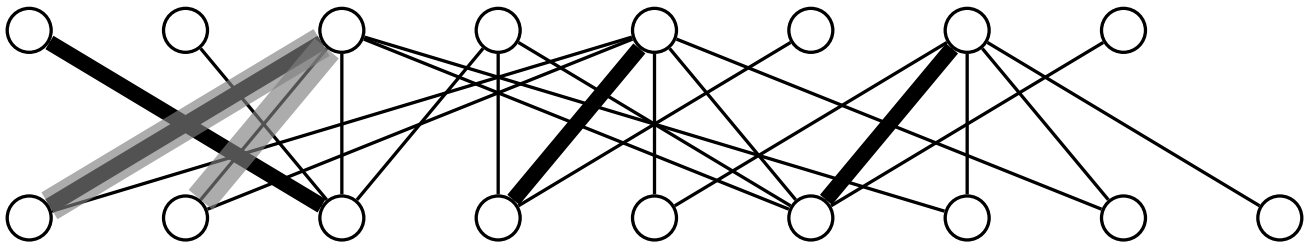
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



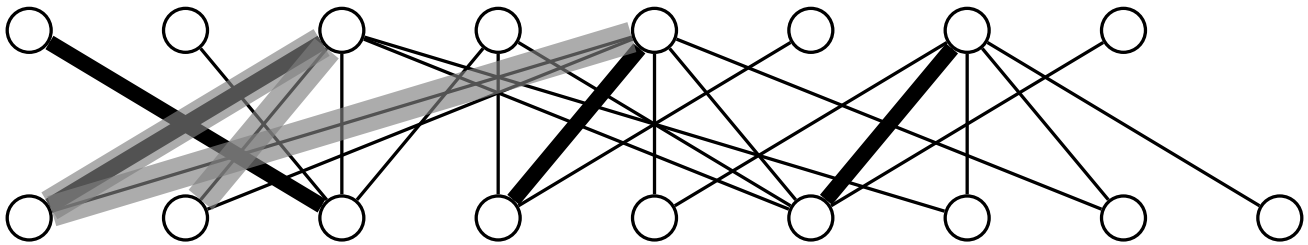
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



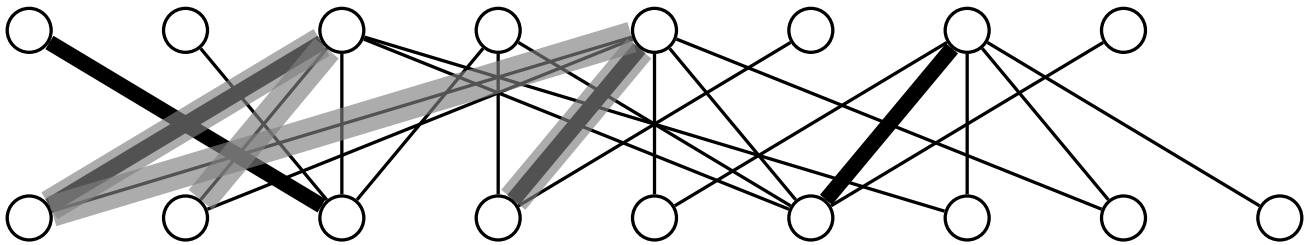
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



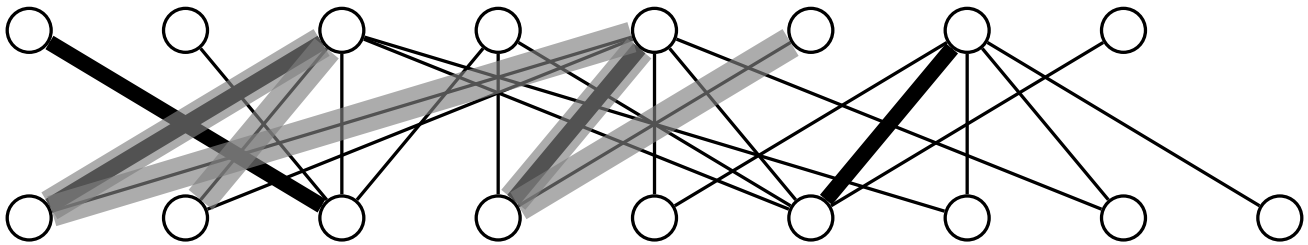
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



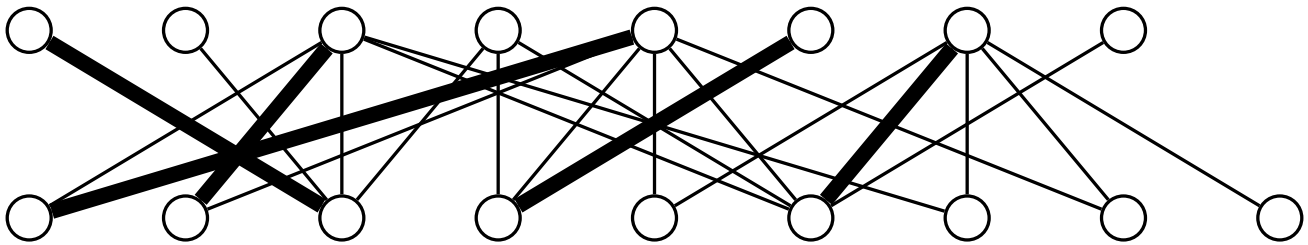
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



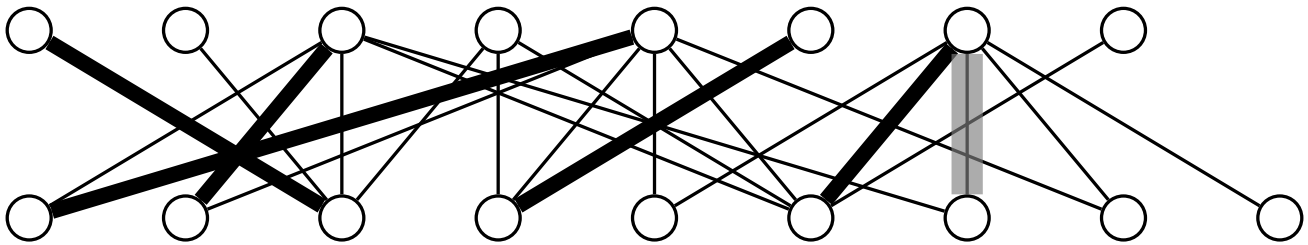
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



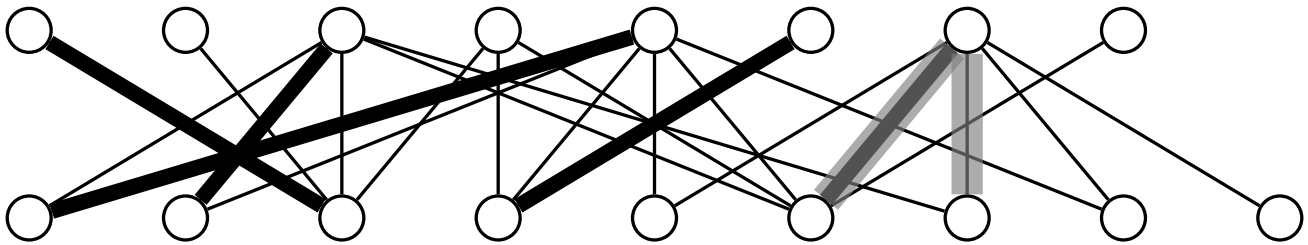
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



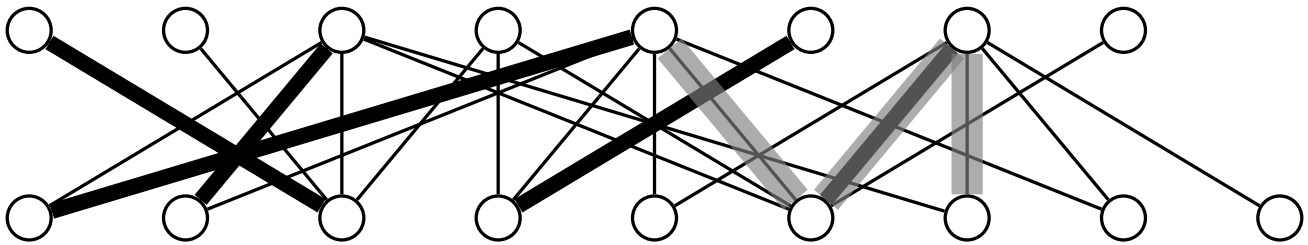
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



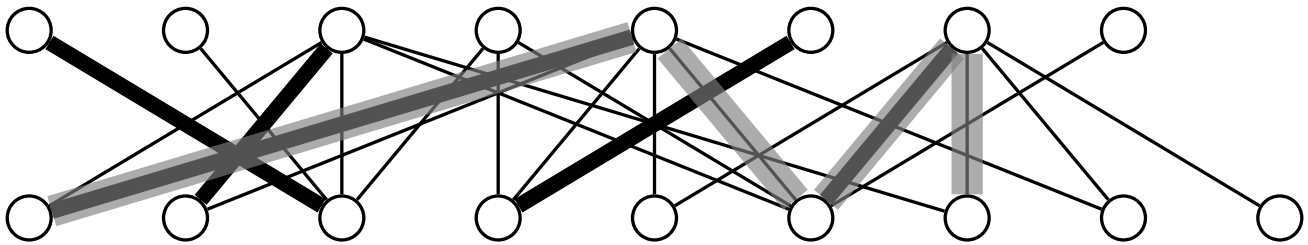
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



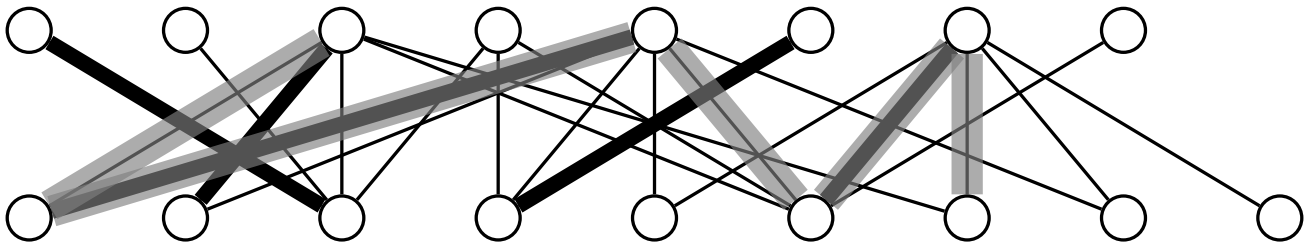
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



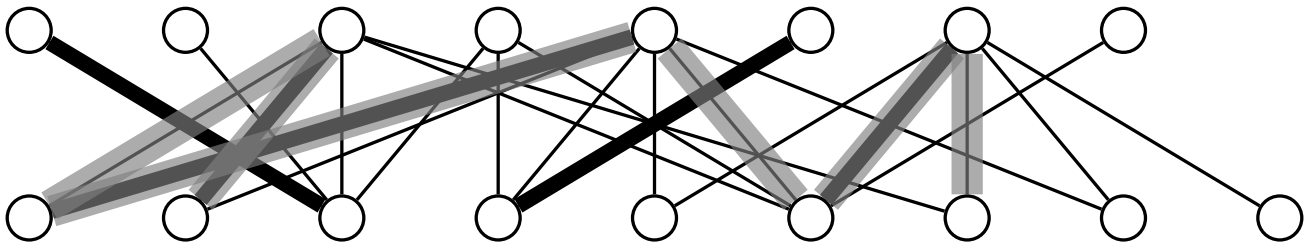
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



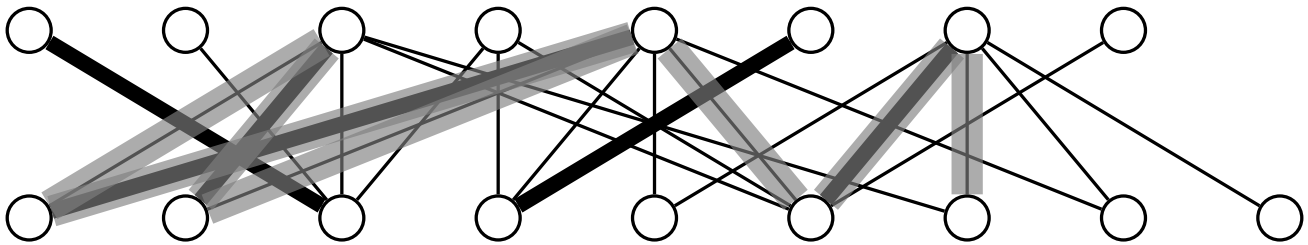
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



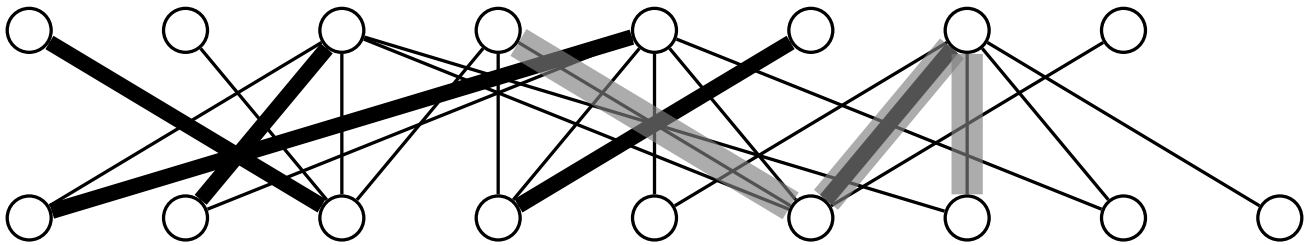
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



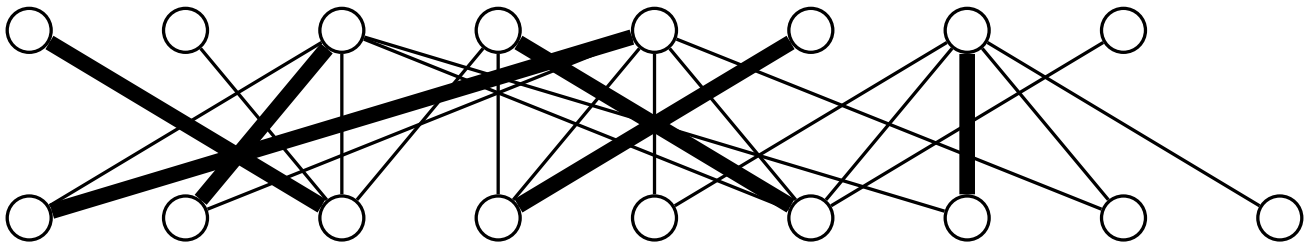
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



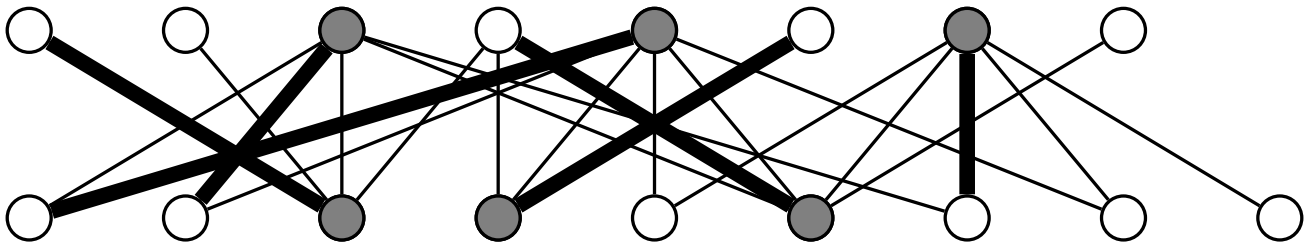
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



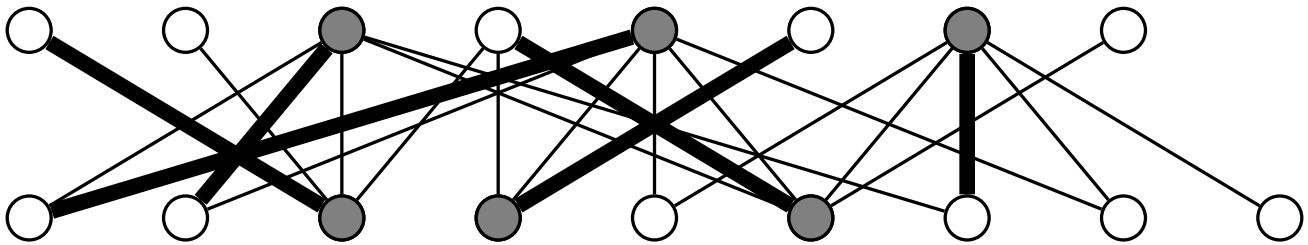
Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Finding a Maximum Matching

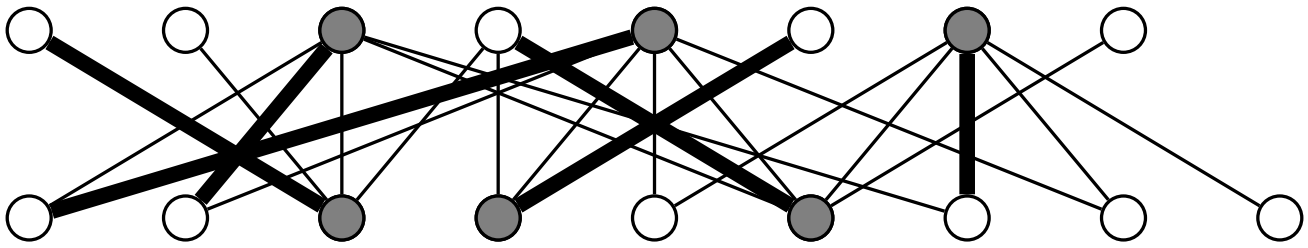
Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.

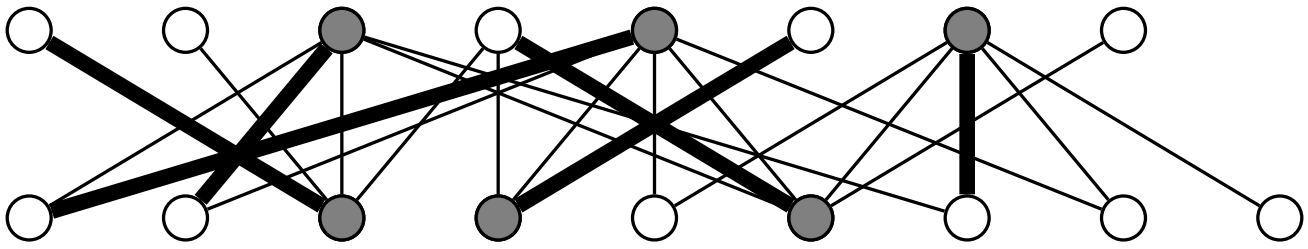


Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.

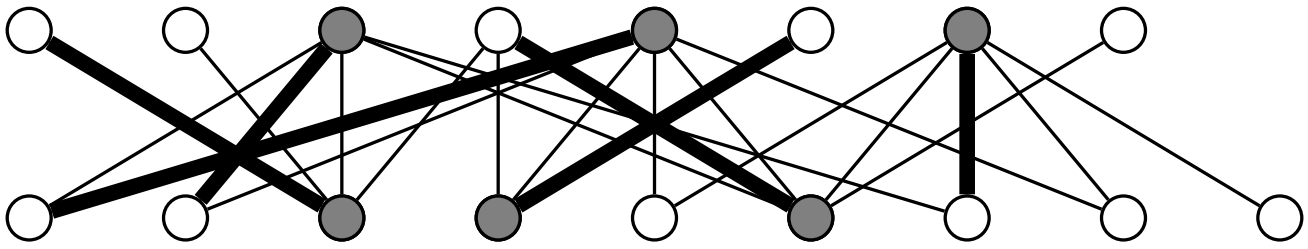


Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



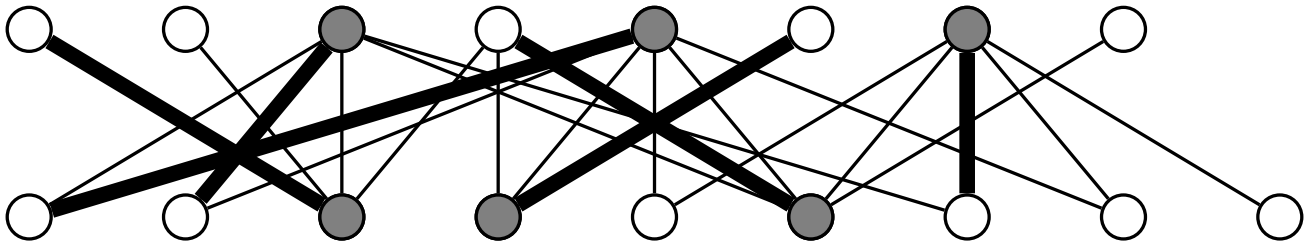
Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

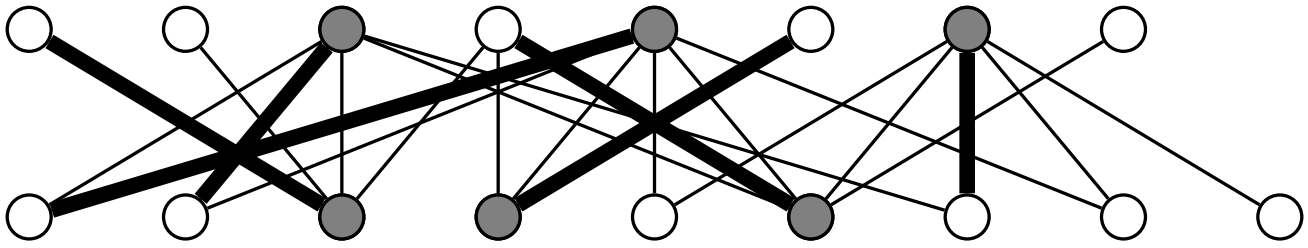
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

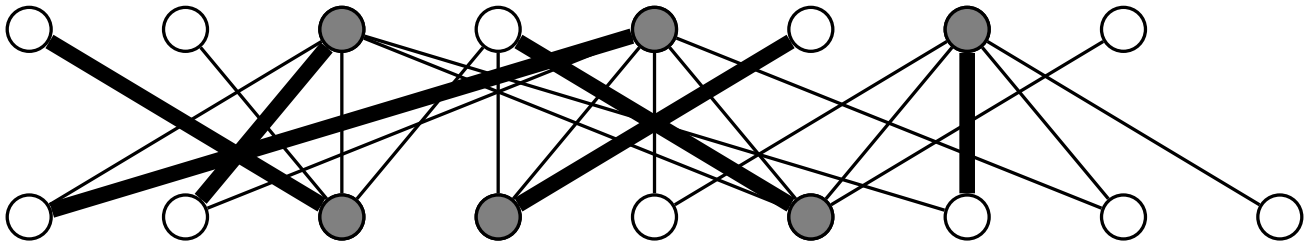
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

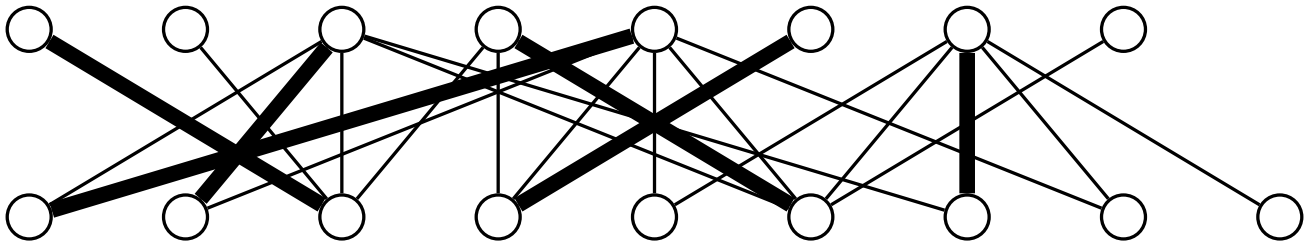
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

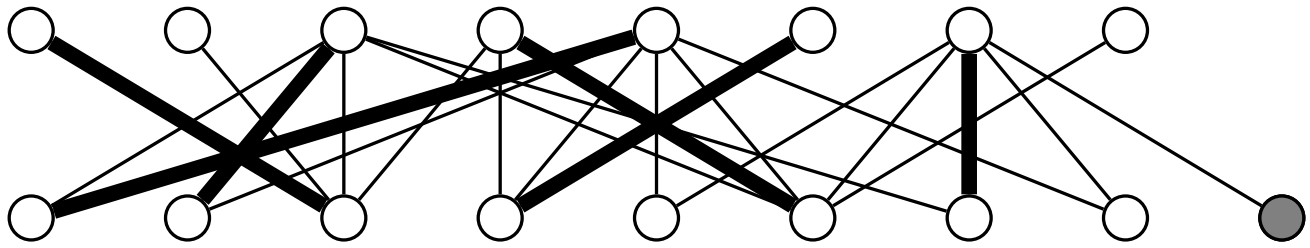
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

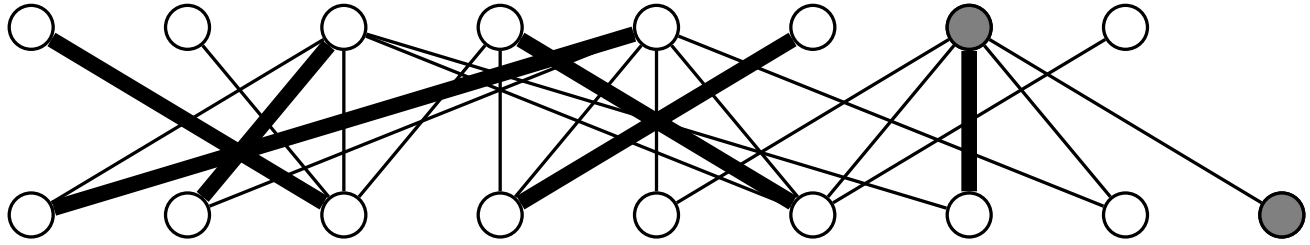
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

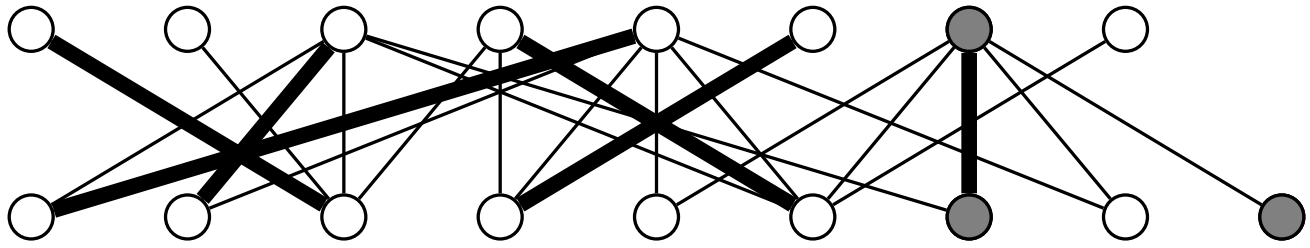
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

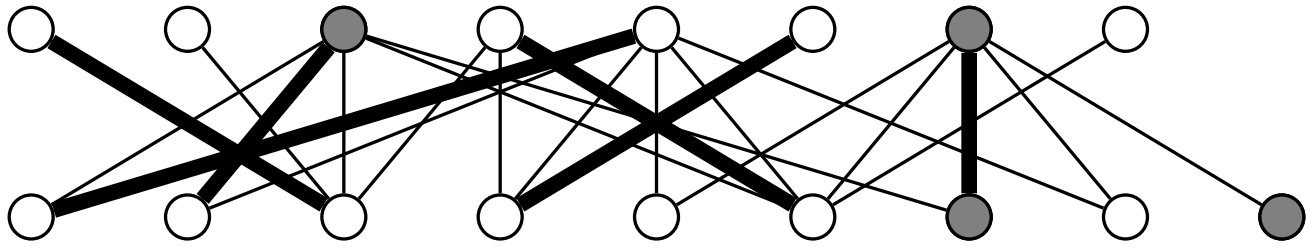
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

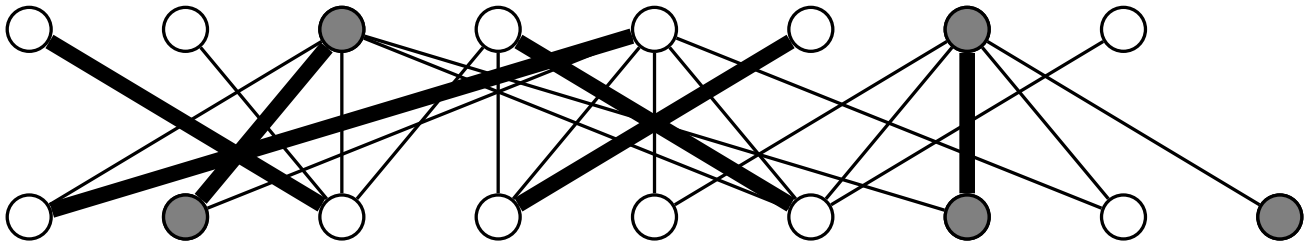
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

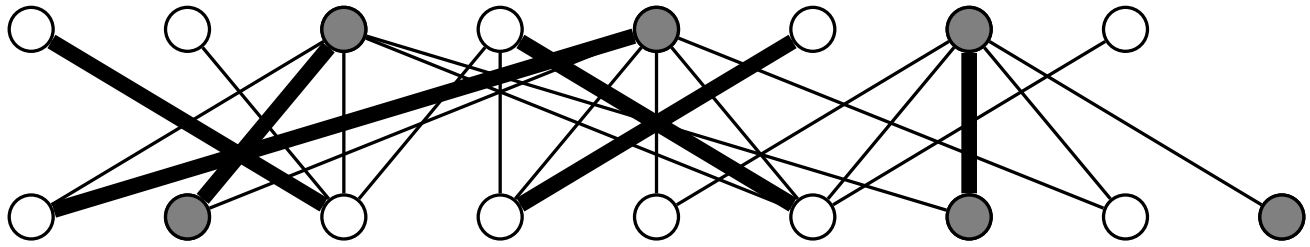
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

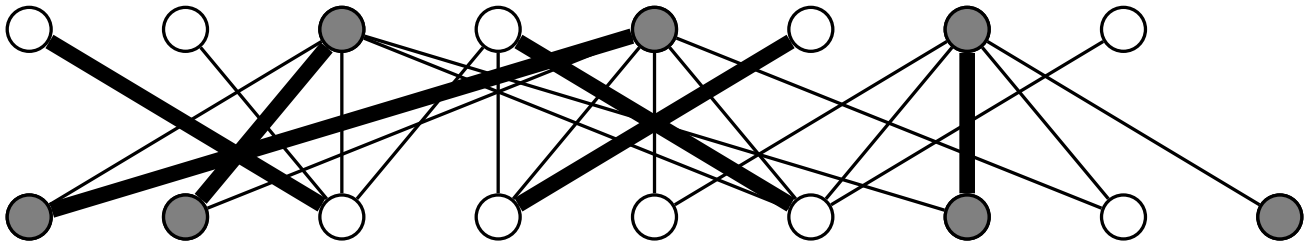
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

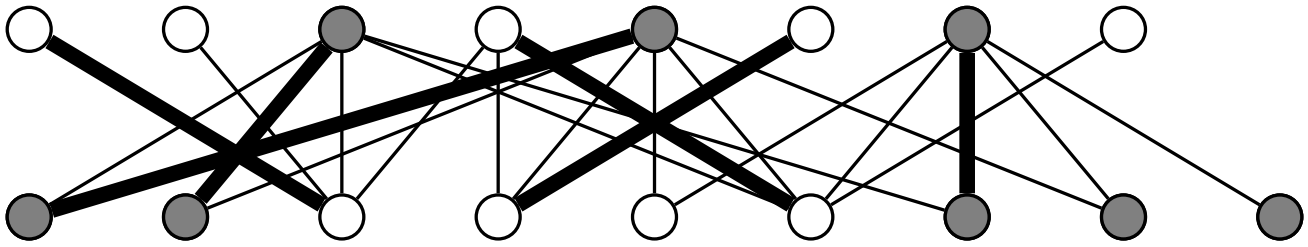
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

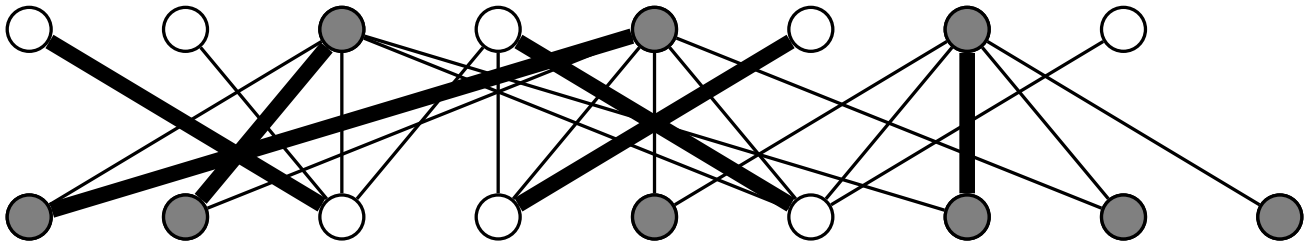
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

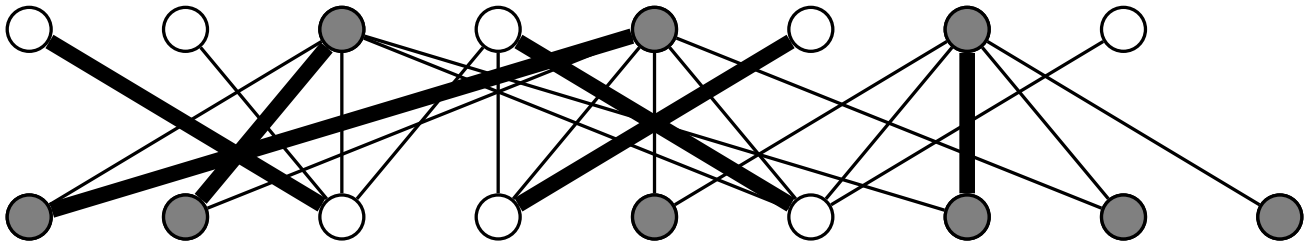
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

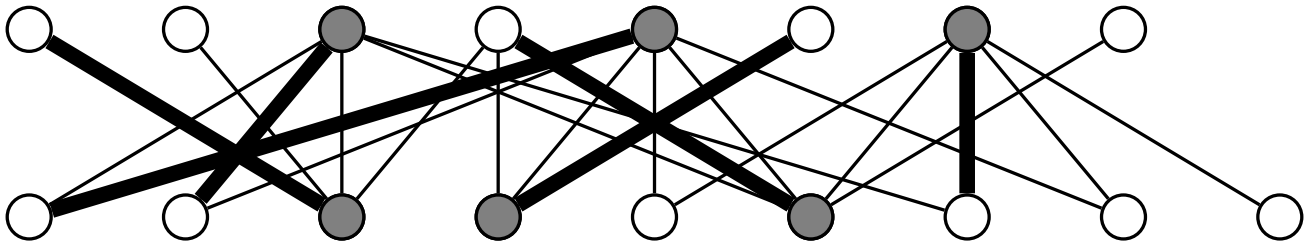
Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

Finding a Maximum Matching

Maximum Matching "Algorithm": Repeatedly find an augmenting path and augment. If you can't, stop.



Definition: A vertex cover is a vertex subset including at least one endpoint of every edge.

Lemma: In every bip. graph, maximum size of a matching equals minimum size of vertex cover. Better way to prove M is maximum!

Question: How do we find a minimum vertex cover?

Answer: Try to find an M-augmenting path, starting from each M-unsaturated vertex in the "friend" side. Keep track of all the vertices Q you reach during exploration. For your vertex cover, take $(\text{Days} \cap Q) \cup (\text{Friends} \setminus Q)$.

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you.

A More General Model

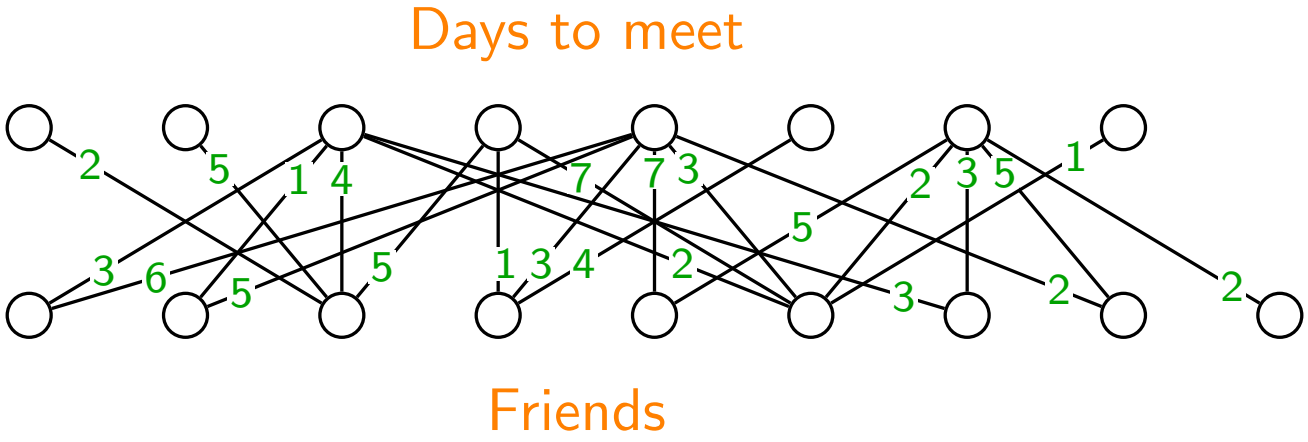
Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting).

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.

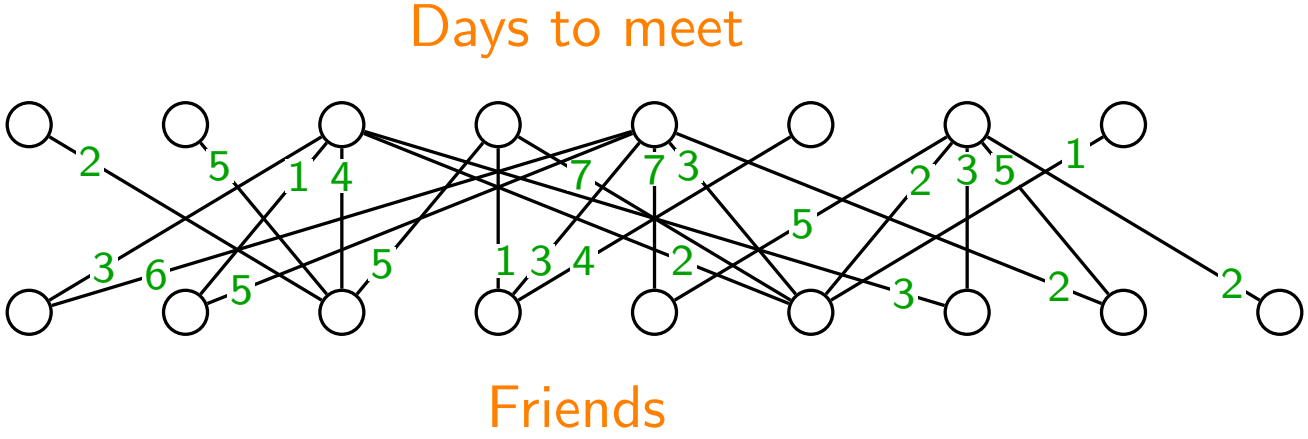
A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.



A More General Model

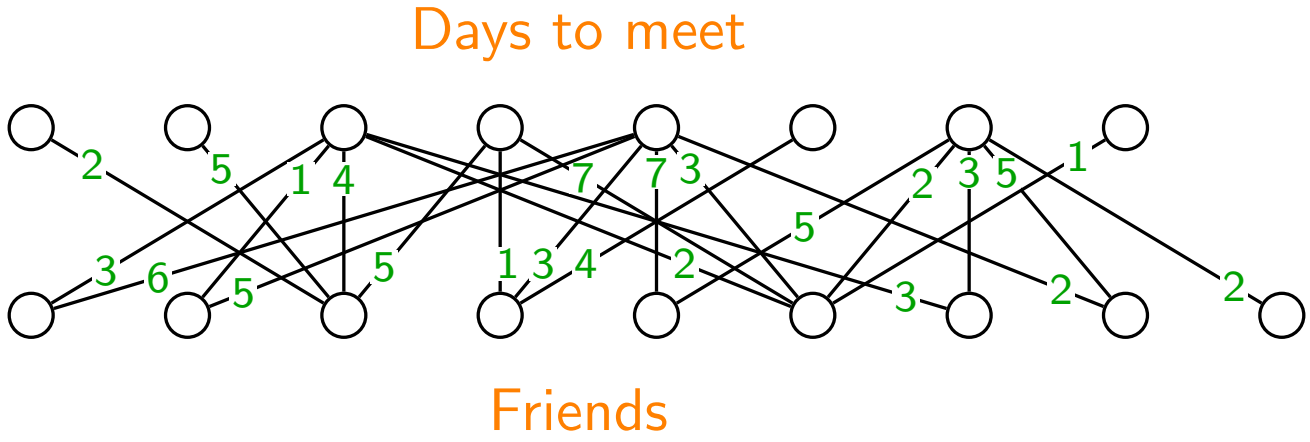
Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.



Many Applications:

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.

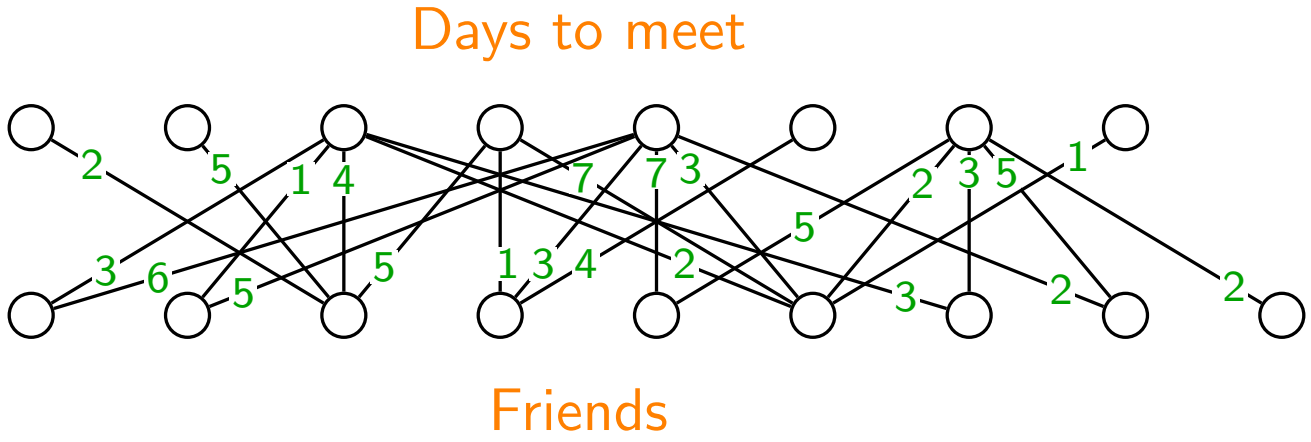


Many Applications:

- Ride Hailing (Uber): minimize waiting time, or max. profit

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.

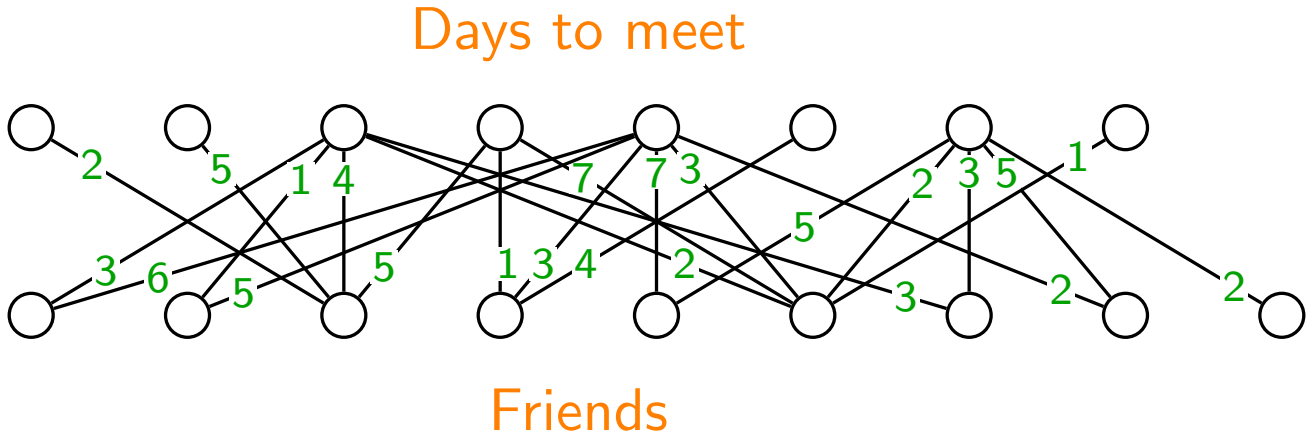


Many Applications:

- Ride Hailing (Uber): minimize waiting time, or max. profit
- Online advertising (AdWords): maximize expected revenue

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.

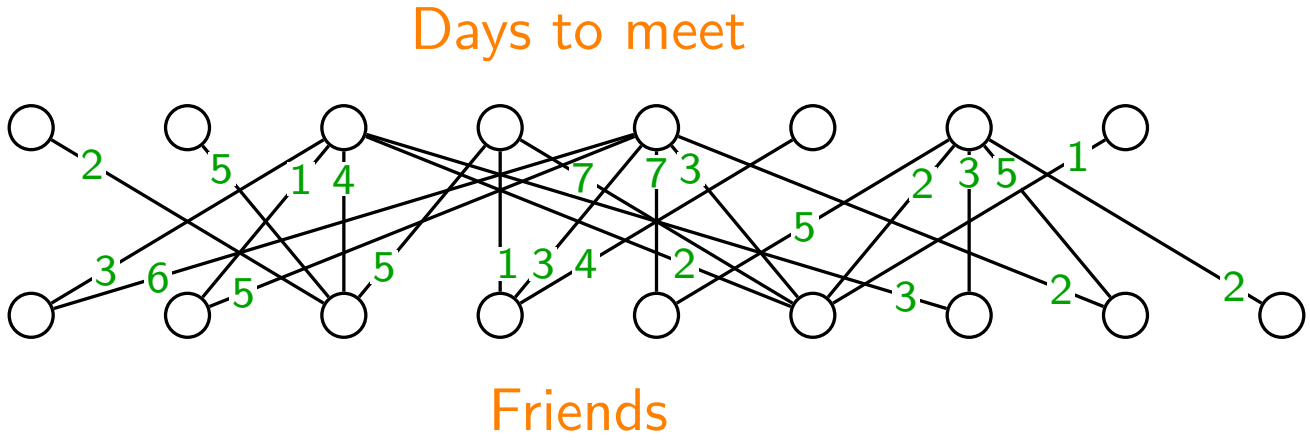


Many Applications:

- Ride Hailing (Uber): minimize waiting time, or max. profit
- Online advertising (AdWords): maximize expected revenue
- E-tail (Amazon): warehouses to customers; min. total distance

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.

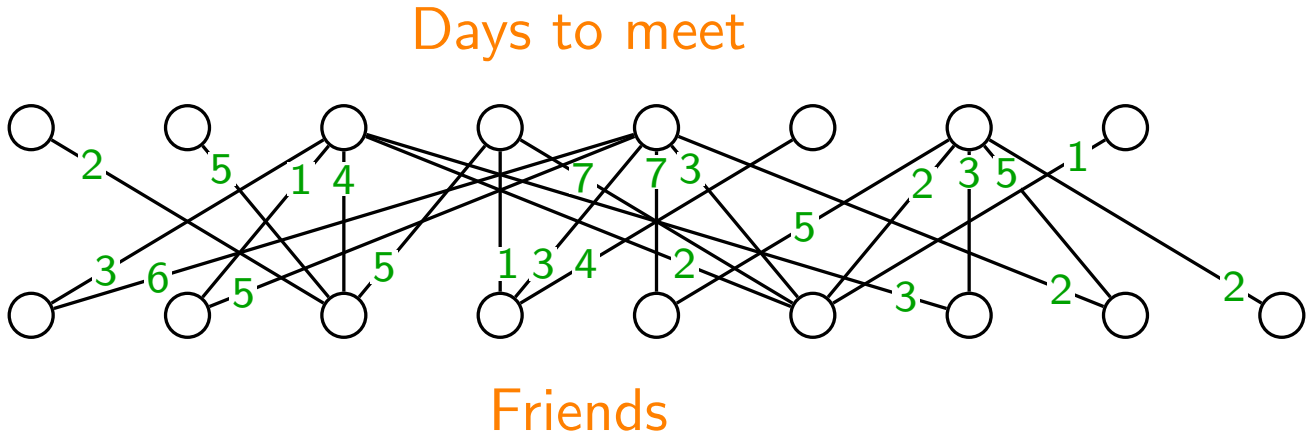


Many Applications:

- Ride Hailing (Uber): minimize waiting time, or max. profit
- Online advertising (AdWords): maximize expected revenue
- E-tail (Amazon): warehouses to customers; min. total distance
- Companies: Match employees to projects, with weights based on alignment of their skills with project needs

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.

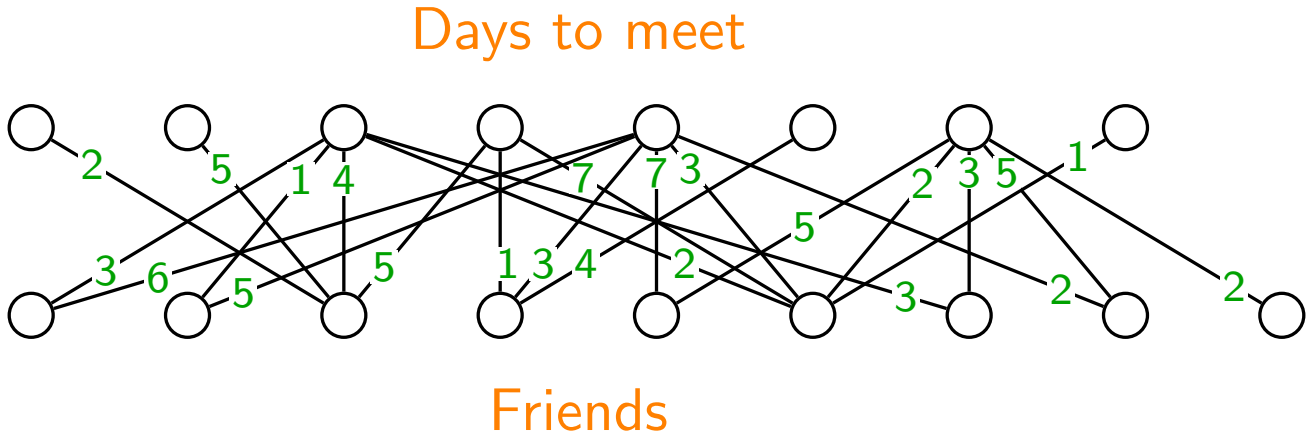


Many Applications:

- Ride Hailing (Uber): minimize waiting time, or max. profit
- Online advertising (AdWords): maximize expected revenue
- E-tail (Amazon): warehouses to customers; min. total distance
- Companies: Match employees to projects, with weights based on alignment of their skills with project needs
- Academic Conferences: match papers with reviewers based on keywords and reviewer expertise

A More General Model

Problem: Not all meet-ups with friends are equally "valuable" to you. Each edge has a weight (how much you value that meeting). Now you want a matching with maximum sum of weights.



Many Applications:

- Ride Hailing (Uber): minimize waiting time, or max. profit
- Online advertising (AdWords): maximize expected revenue
- E-tail (Amazon): warehouses to customers; min. total distance
- Companies: Match employees to projects, with weights based on alignment of their skills with project needs
- Academic Conferences: match papers with reviewers based on keywords and reviewer expertise

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched. For G with edge weights $w : E(G) \to \mathbb{R}$, a vertex cover is a function $f : V(G) \to \mathbb{R}$ such that $w(xy) \leqslant f(x) + f(y)$ for each edge xy.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched. For G with edge weights $w : E(G) \to \mathbb{R}$, a vertex cover is a function $f : V(G) \to \mathbb{R}$ such that $w(xy) \leqslant f(x) + f(y)$ for each edge xy.

Key Lemma: For a perfect matching M in G with weights w, and a weighted cover f, we always have $\sum_{e \in M} w(e) \leqslant \sum_{v \in V(G)} f(v)$. This inequality holds with equality iff M is a maximum weight matching and f is a minimum weight vertex cover.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched. For G with edge weights $w : E(G) \to \mathbb{R}$, a vertex cover is a function $f : V(G) \to \mathbb{R}$ such that $w(xy) \leqslant f(x) + f(y)$ for each edge xy.

Key Lemma: For a perfect matching M in G with weights w, and a weighted cover f, we always have $\sum_{e \in M} w(e) \leqslant \sum_{v \in V(G)} f(v)$. This inequality holds with equality iff M is a maximum weight matching and f is a minimum weight vertex cover.
Proof: Sum $w(xy) \leqslant f(x) + f(y)$ over all $xy \in M$.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched. For G with edge weights $w : E(G) \to \mathbb{R}$, a vertex cover is a function $f : V(G) \to \mathbb{R}$ such that $w(xy) \leqslant f(x) + f(y)$ for each edge xy.

Key Lemma: For a perfect matching M in G with weights w, and a weighted cover f, we always have $\sum_{e \in M} w(e) \leqslant \sum_{v \in V(G)} f(v)$. This inequality holds with equality iff M is a maximum weight matching and f is a minimum weight vertex cover.

Proof: Sum $w(xy) \leqslant f(x) + f(y)$ over all $xy \in M$. If we have equality over the sum, then equality holds for each edge.

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched. For G with edge weights $w : E(G) \to \mathbb{R}$, a vertex cover is a function $f : V(G) \to \mathbb{R}$ such that $w(xy) \leqslant f(x) + f(y)$ for each edge xy.

Key Lemma: For a perfect matching M in G with weights w, and a weighted cover f, we always have $\sum_{e \in M} w(e) \leqslant \sum_{v \in V(G)} f(v)$. This inequality holds with equality iff M is a maximum weight matching and f is a minimum weight vertex cover.

Proof: Sum $w(xy) \leqslant f(x) + f(y)$ over all $xy \in M$. If we have equality over the sum, then equality holds for each edge. ■

Maximum Weight Matchings: The Setup

Assumptions:

- Numbers of friends and days are equal. (if not, add some)
- No negative weights. (if not, delete weight negative edges)
- All possible edges from friends to days. (if not, add them)

Definitions: In a perfect matching, every vertex is matched. For G with edge weights $w : E(G) \to \mathbb{R}$, a vertex cover is a function $f : V(G) \to \mathbb{R}$ such that $w(xy) \leqslant f(x) + f(y)$ for each edge xy.

Key Lemma: For a perfect matching M in G with weights w, and a weighted cover f, we always have $\sum_{e \in M} w(e) \leqslant \sum_{v \in V(G)} f(v)$. This inequality holds with equality iff M is a maximum weight matching and f is a minimum weight vertex cover.
Proof: Sum $w(xy) \leqslant f(x) + f(y)$ over all $xy \in M$. If we have equality over the sum, then equality holds for each edge. ■

Key Theorem: Fix a bipartite G with edge weights w. For every max. perfect matching M, there is a min. vertex cover f such that $\sum_{xy \in M} w(xy) = \sum_{v \in V(G)} f(v)$. Proves f and M are both optimal.

Finding a Max Weight Matching and Min Weight Cover

Finding a Max Weight Matching and Min Weight Cover

Definition: For a graph G with weights w and a vertex cover f, the equality subgraph G_f is the (unweighted) graph with all the edges xy such that $w(xy) = f(x) + f(y)$.

Finding a Max Weight Matching and Min Weight Cover

Definition: For a graph G with weights w and a vertex cover f, the equality subgraph G_f is the (unweighted) graph with all the edges xy such that $w(xy) = f(x) + f(y)$.

Maximum Weight Matching "Algorithm":

1. Start with some vertex cover f such that G_f has edges.

Finding a Max Weight Matching and Min Weight Cover

Definition: For a graph G with weights w and a vertex cover f, the equality subgraph G_f is the (unweighted) graph with all the edges xy such that $w(xy) = f(x) + f(y)$.

Maximum Weight Matching "Algorithm":

1. Start with some vertex cover f such that G_f has edges.
2. Find a maximum matching M and min vertex cover Q in G_f.

Finding a Max Weight Matching and Min Weight Cover

Definition: For a graph G with weights w and a vertex cover f, the equality subgraph G_f is the (unweighted) graph with all the edges xy such that $w(xy) = f(x) + f(y)$.

Maximum Weight Matching "Algorithm":

1. Start with some vertex cover f such that G_f has edges.
2. Find a maximum matching M and min vertex cover Q in G_f.
3. If M is perfect in G, then halt and output M and Q.

Finding a Max Weight Matching and Min Weight Cover

Definition: For a graph G with weights w and a vertex cover f, the equality subgraph G_f is the (unweighted) graph with all the edges xy such that $w(xy) = f(x) + f(y)$.

Maximum Weight Matching "Algorithm":

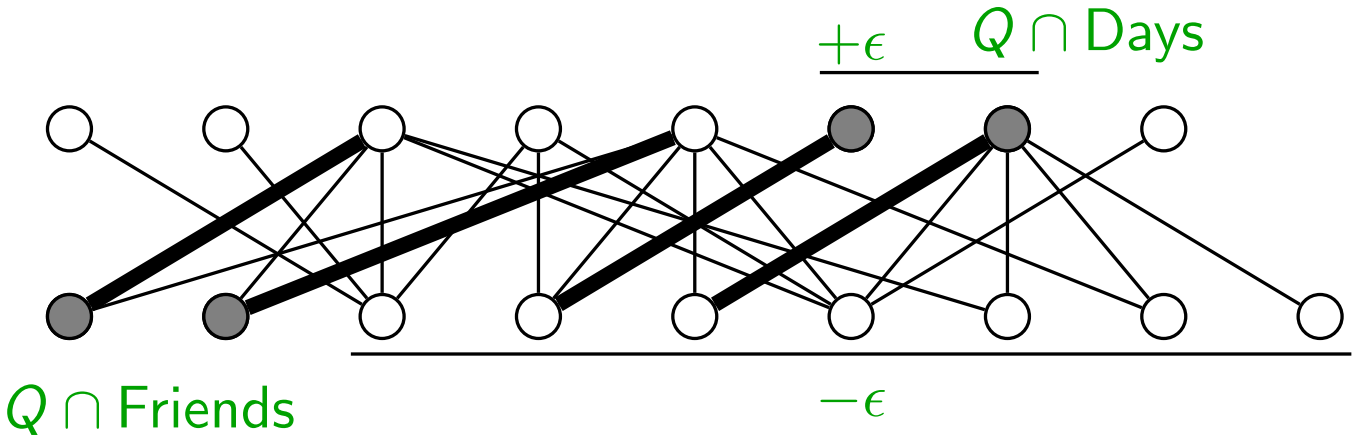
1. Start with some vertex cover f such that G_f has edges.
2. Find a maximum matching M and min vertex cover Q in G_f.
3. If M is perfect in G, then halt and output M and Q.
4. Modify f to get f' with $\sum_{v \in V(G)} f'(v) < \sum_{v \in V(G)} f(v)$. That is, f' is a smaller cover. Go to 2.

Finding a Max Weight Matching and Min Weight Cover

Definition: For a graph G with weights w and a vertex cover f, the equality subgraph G_f is the (unweighted) graph with all the edges xy such that $w(xy) = f(x) + f(y)$.

Maximum Weight Matching "Algorithm":

1. Start with some vertex cover f such that G_f has edges.
2. Find a maximum matching M and min vertex cover Q in G_f.
3. If M is perfect in G, then halt and output M and Q.
4. Modify f to get f' with $\sum_{v \in V(G)} f'(v) < \sum_{v \in V(G)} f(v)$. That is, f' is a smaller cover. Go to 2.



Recap:

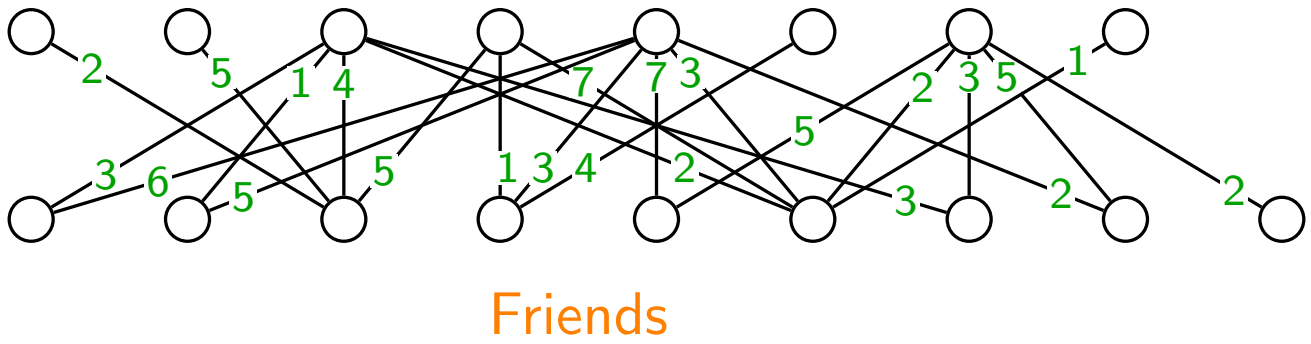
Problem: You want the best possible matching between friends and days;

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.

Recap:

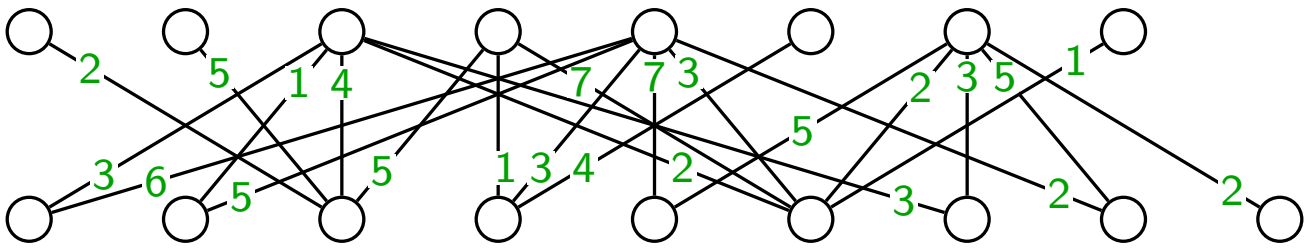
Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.



Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.

Days to meet



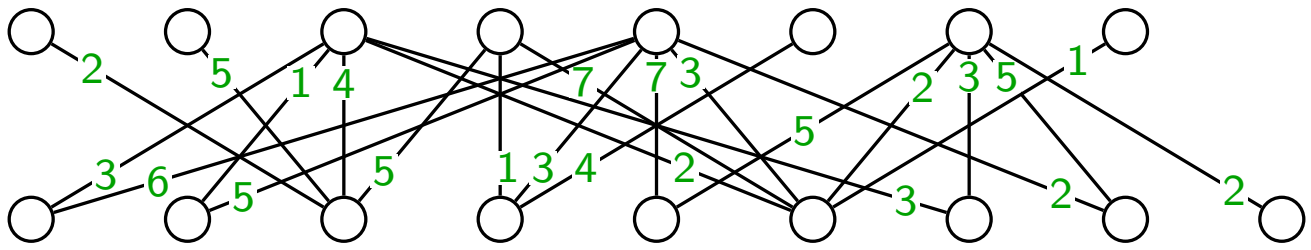
Friends

Obs (Weak Duality): Min weight vertex cover gives upper bound.

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.

Days to meet



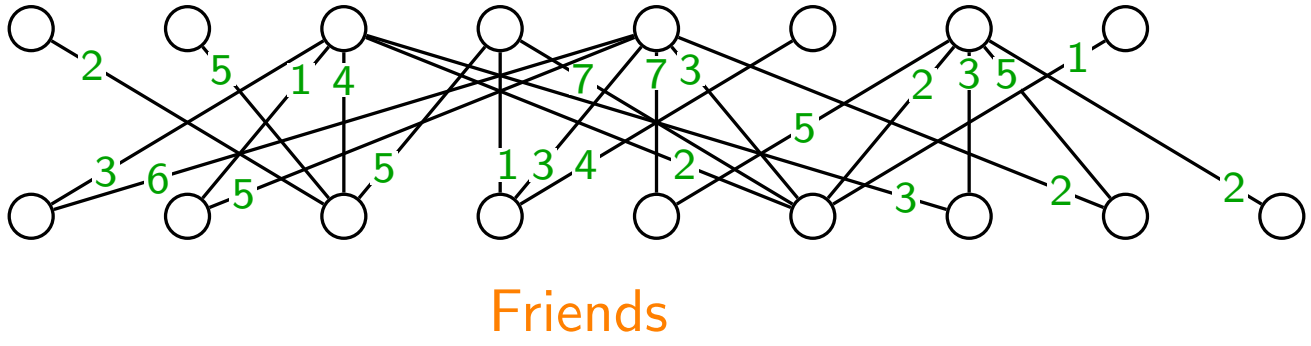
Friends

Obs (Weak Duality): Min weight vertex cover gives upper bound.

Key Thm (Strong Duality): Inequality always holds with equality.

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.

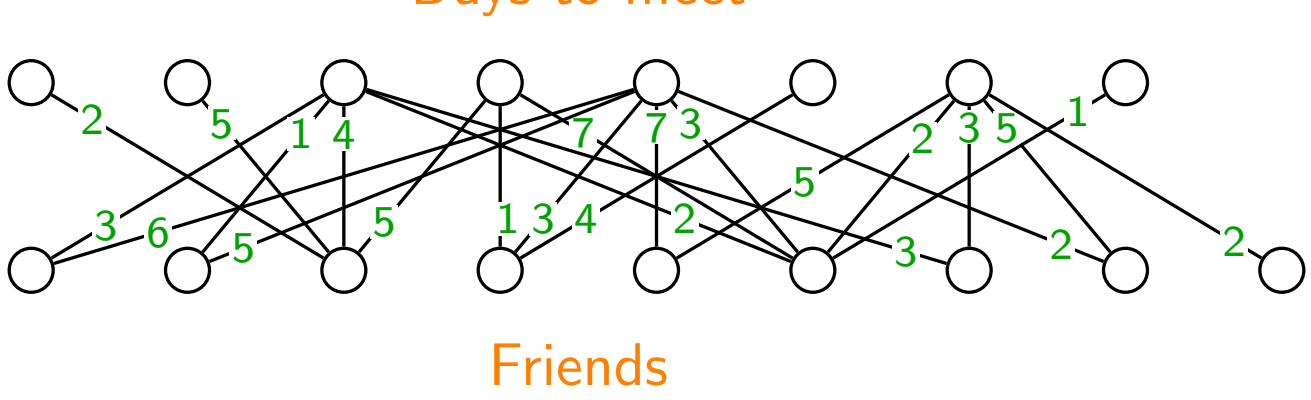


Obs (Weak Duality): Min weight vertex cover gives upper bound.
Key Thm (Strong Duality): Inequality always holds with equality.

Maximum (Weight) Matching Algorithm:

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.



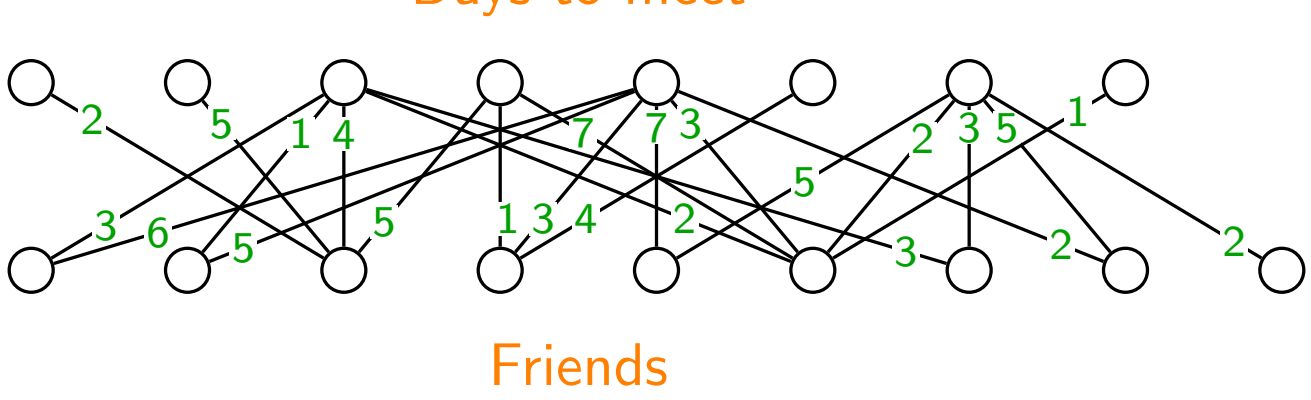
Obs (Weak Duality): Min weight vertex cover gives upper bound.
Key Thm (Strong Duality): Inequality always holds with equality.

Maximum (Weight) Matching Algorithm:

1. Start with a solution M and work to improve it

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.



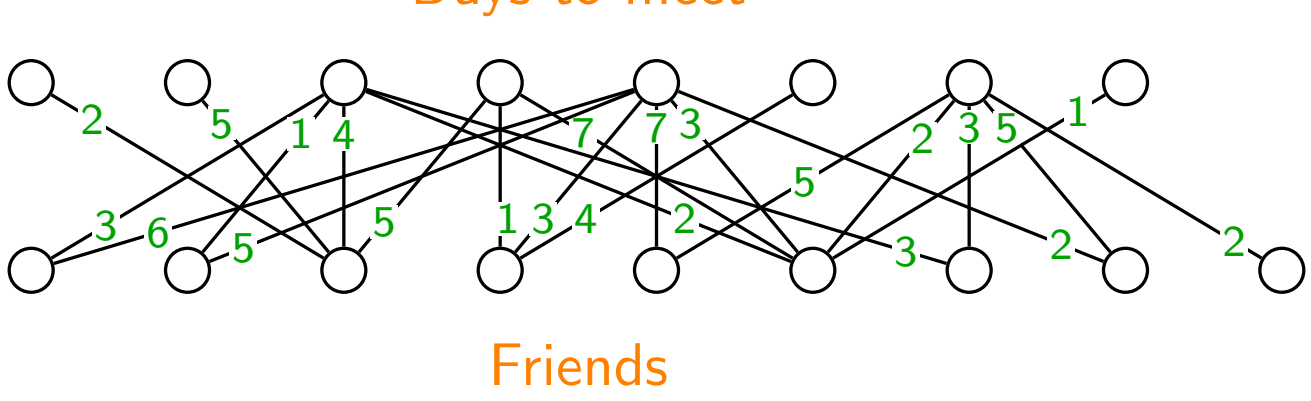
Obs (Weak Duality): Min weight vertex cover gives upper bound.
Key Thm (Strong Duality): Inequality always holds with equality.

Maximum (Weight) Matching Algorithm:

1. Start with a solution M and work to improve it
   - unweighted: M-augmenting paths

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.



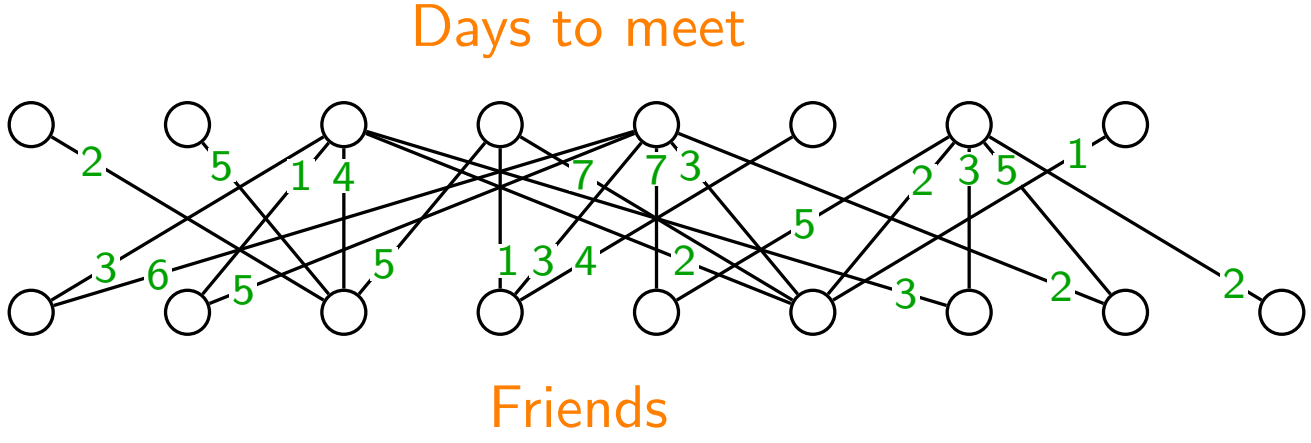
Obs (Weak Duality): Min weight vertex cover gives upper bound.
Key Thm (Strong Duality): Inequality always holds with equality.

Maximum (Weight) Matching Algorithm:

1. Start with a solution M and work to improve it
   - unweighted: M-augmenting paths
   - weighted: Adjustment to f

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.



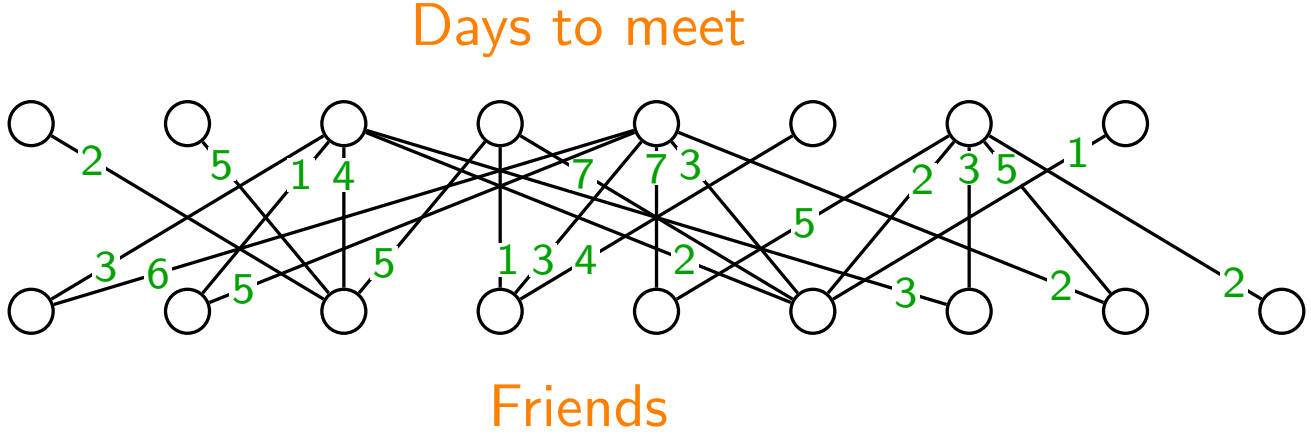
Obs (Weak Duality): Min weight vertex cover gives upper bound.
Key Thm (Strong Duality): Inequality always holds with equality.

Maximum (Weight) Matching Algorithm:

1. Start with a solution M and work to improve it
   - unweighted: M-augmenting paths
   - weighted: Adjustment to f
2. When we get stuck improving, we output matching M, and cover Q to show M is optimal.

Recap:

Problem: You want the best possible matching between friends and days; model as maximum (weight) bipartite matching.



Obs (Weak Duality): Min weight vertex cover gives upper bound.
Key Thm (Strong Duality): Inequality always holds with equality.

Maximum (Weight) Matching Algorithm:

1. Start with a solution M and work to improve it
   - unweighted: M-augmenting paths
   - weighted: Adjustment to f
2. When we get stuck improving, we output matching M, and cover Q to show M is optimal.

Finally: Lots more applications, in various fields!