Finding a Maximum Matching in Non-Bipartite Graphs

Alicia Thilani Singham Goodwin

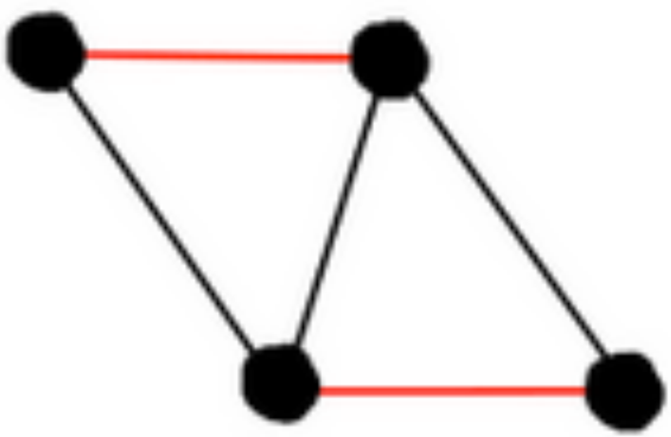
18.304 • 3/22/2013

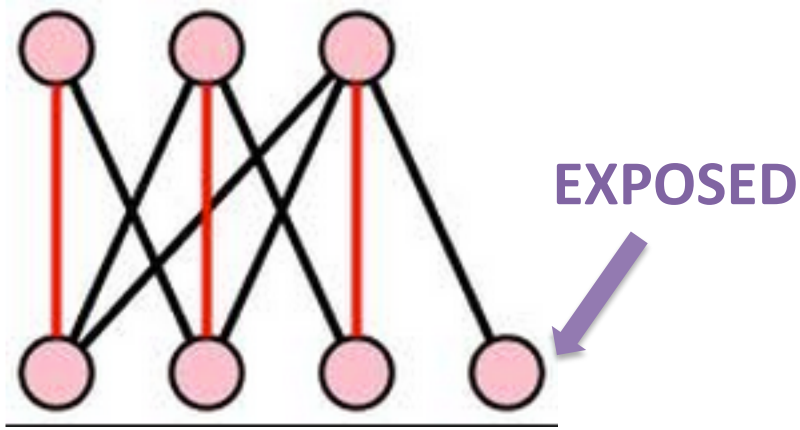
Our Goal

- Make an algorithm to find the **largest cardinality matching** (most sets of partners) in ANY graph.
- *Method: Generalize the maximum matching algorithm for bipartite graphs*

Review (1/2)

- **Matching:** a set of edges without common vertices

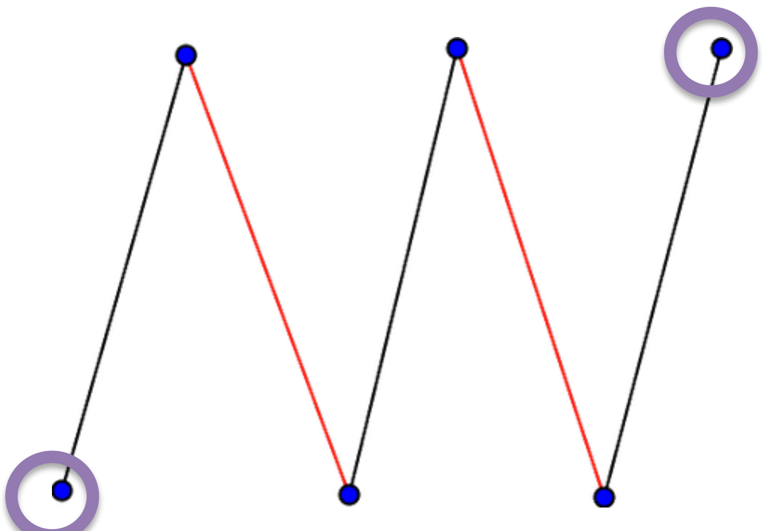




  - **Maximum Cardinality Matching:** largest # of edges

Review (2/2)

- An **alternating path** with respect to **M** alternates between edges in **M** and in **E-M**



- An **augmenting path** with respect to **M** is an alternating path with **first and last vertices exposed**

Bipartite Graph Algorithm

1 – Start with any matching **M** (let's say M = {})

2 – As long as there exists an **augmenting path** with respect to **M**:

  3 – Find augmenting path **P** with respect to **M**

  4 – Augment **M** along **P**: **M' = M Δ P**

  5 – Replace **M** with the new **M'**

Bipartite Graph Algorithm

1 – Start with any matching **M** (let's say M = {})

2 – As long as there exists an **augmenting path** with respect to **M**:

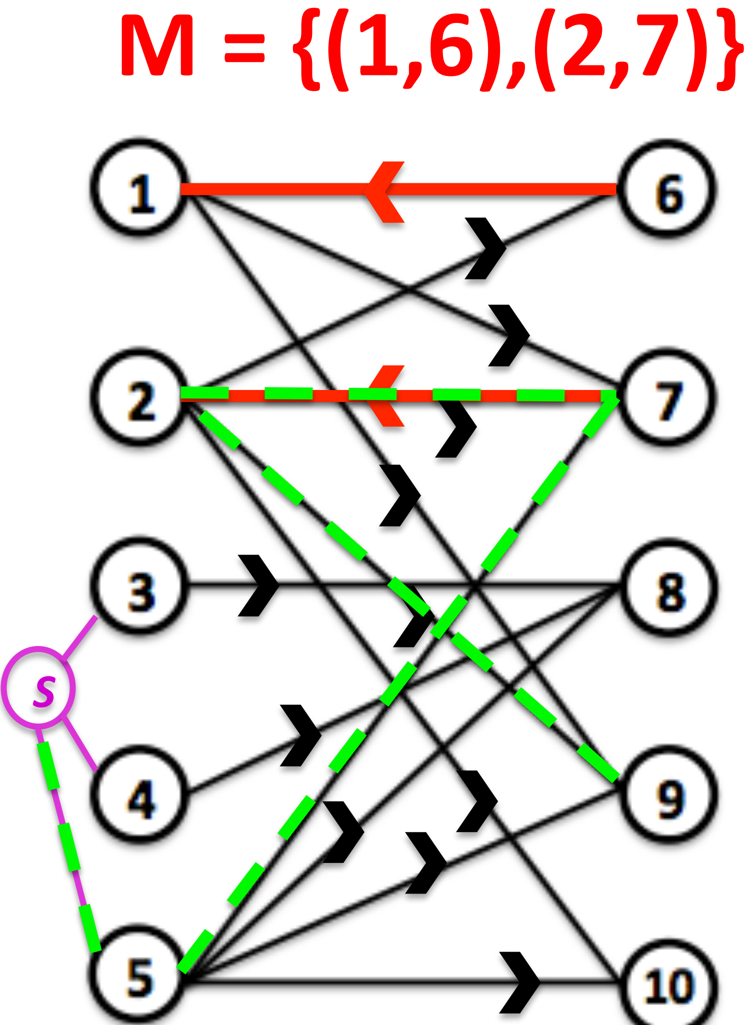
3 – Find augmenting path P with respect to M

4 – Augment **M** along **P**: **M' = M Δ P**

5 – Replace **M** with the new **M'**

HOW???

Bipartite: Finding an Augmenting Path



1 – Direct all edges in the matching from B to A, and all edges not in the matching from A to B

2 – Create a node *s* that connects to all exposed vertices in set A

3 – Do a Breadth First Search to find an exposed vertex in set B from node *s*

Same

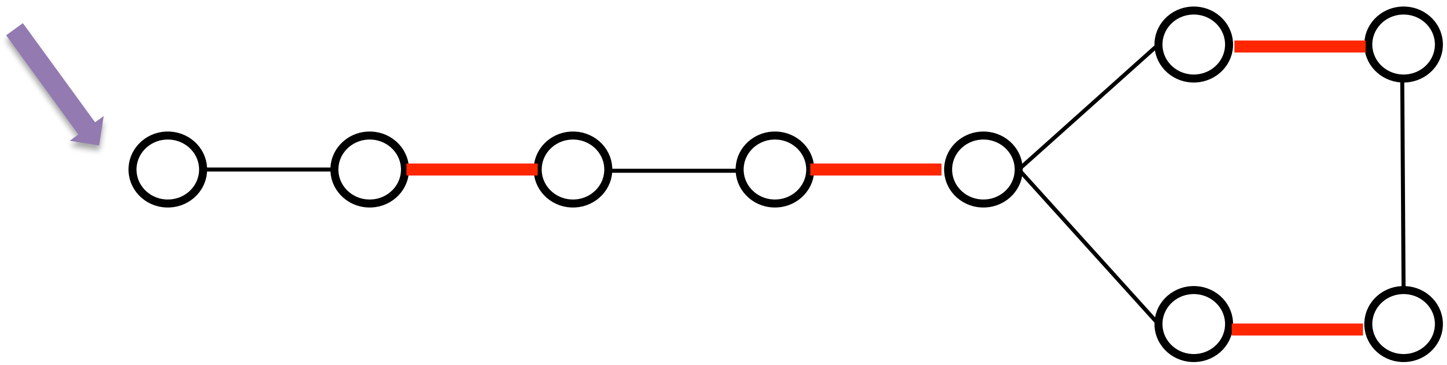
- A matching is maximum **if and only if** there are no augmenting paths
- *General Plan:* keep looking for augmenting paths to expand the matching

Different

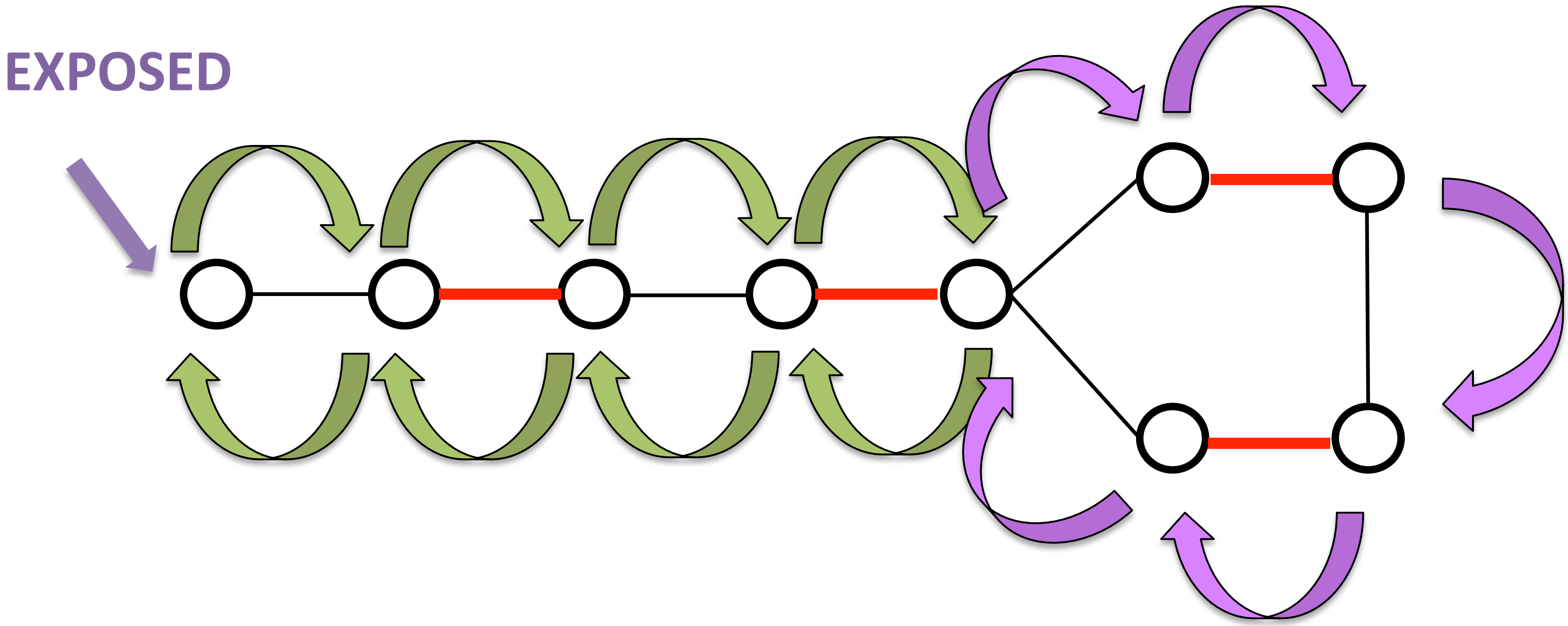
- We can't add direction to the edges to find augmenting paths
- We might find "fake" augmenting paths, called FLOWERS

Flowers: Stems & Blossoms

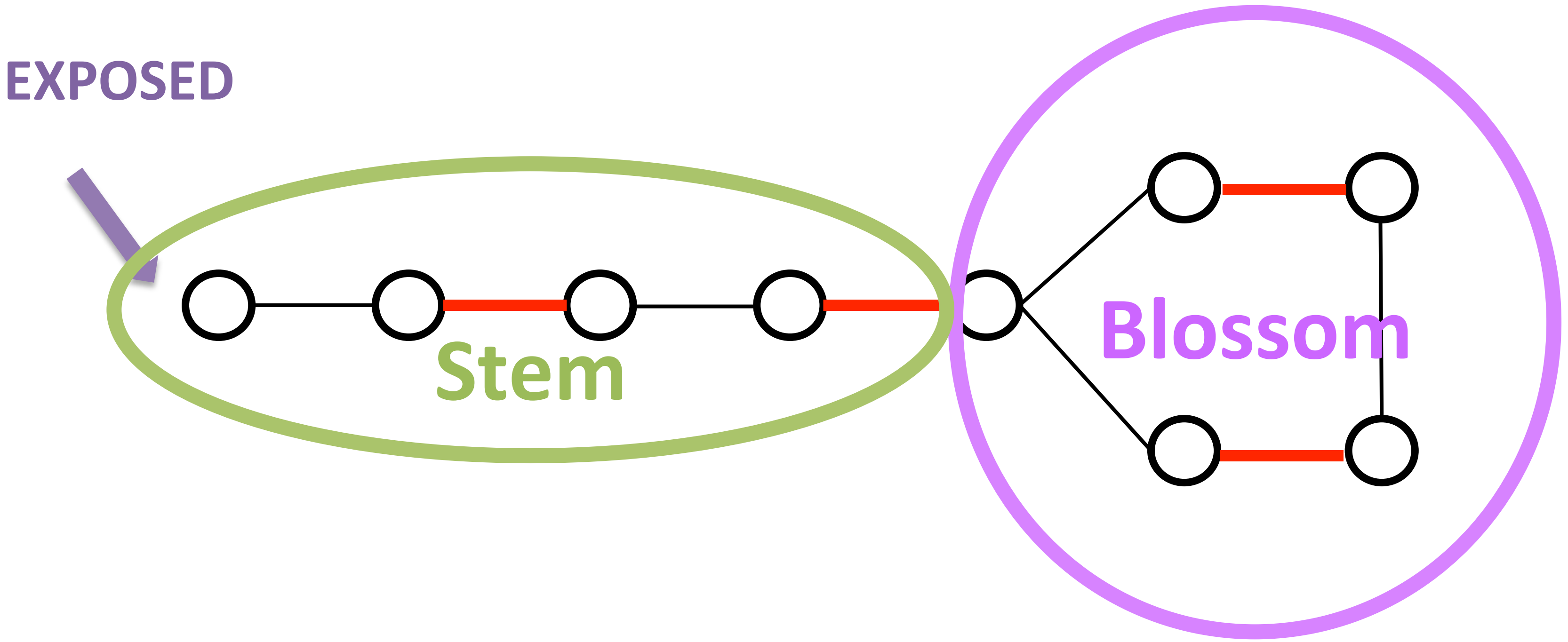
EXPOSED



Flowers: Stems & Blossoms

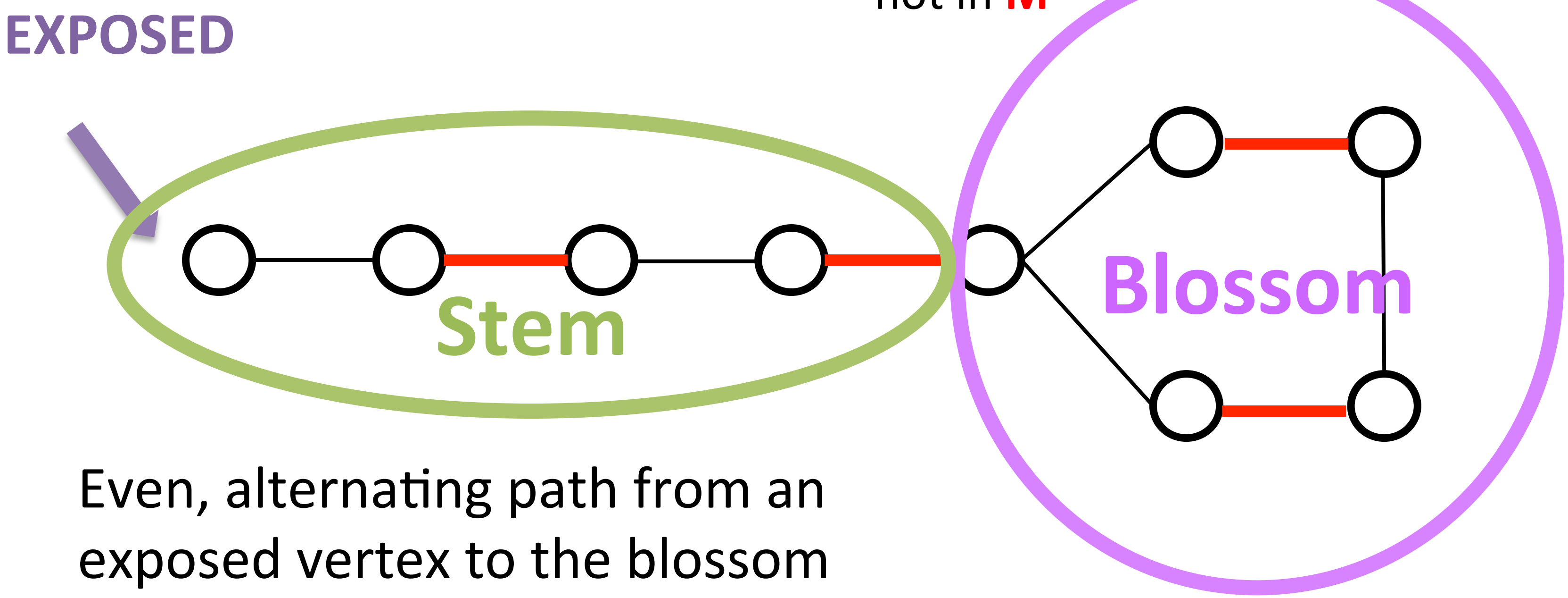


Flowers: Stems & Blossoms



Flowers: Stems & Blossoms

Odd, alternating cycle with two edges adjacent to the stem and not in **M**



Even, alternating path from an exposed vertex to the blossom

Revised Algorithm

1 – Start with any matching **M** (let's say M = {})

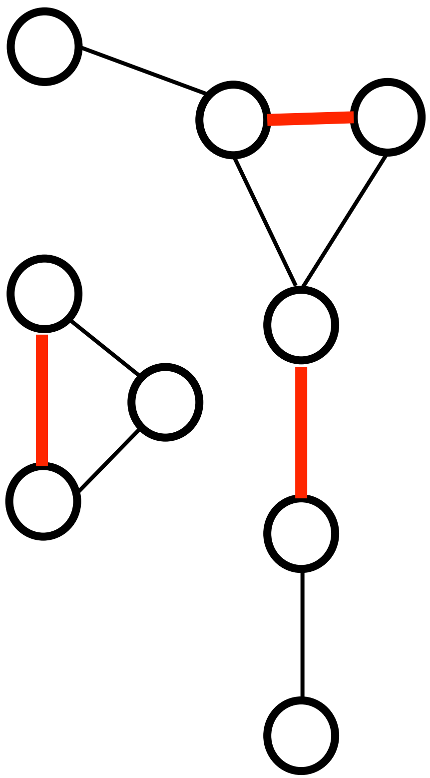
2 – Find a **flower**, **augmenting path** or **neither**:

   3 – If **neither**: We're done!

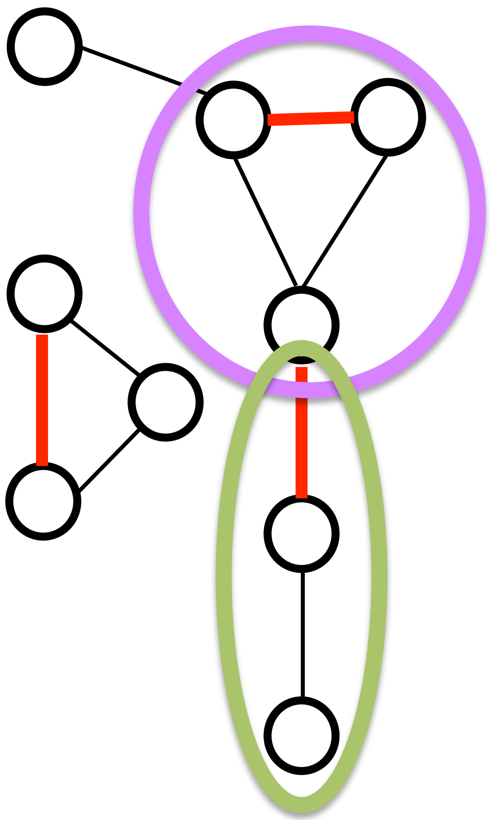
   4 – If **augmenting path**: augment to **M' = M Δ P**

   5 – If **flower**: find a larger matching or decide that **M** is maximum…

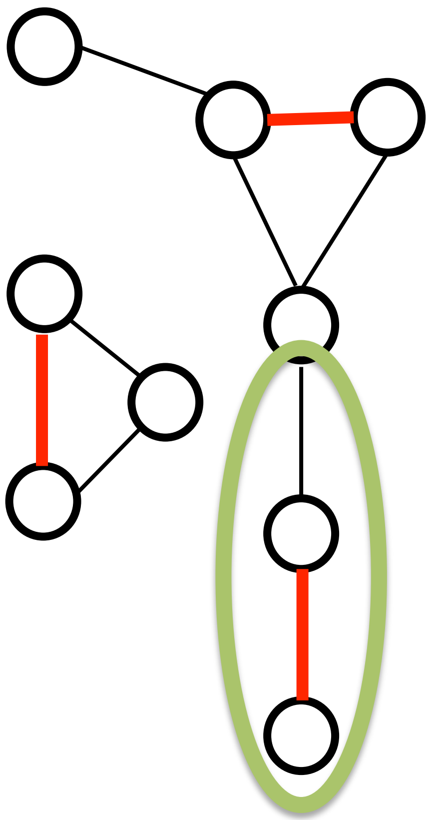
<u>Theorem</u>: M is maximum in G if and only if M-B is maximum in G-B



Theorem: M is maximum in G if and only if M-B is maximum in G-B



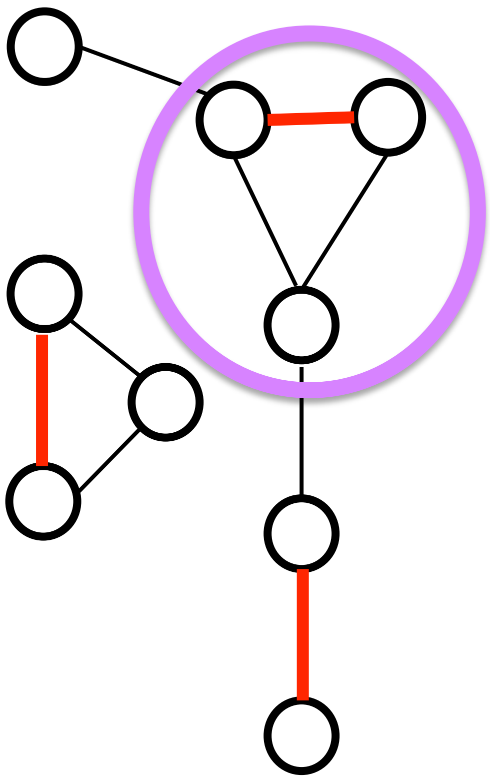
<u>Theorem</u>: M is maximum in G if and only if M-B is maximum in G-B



1- Flip the stem

- -The matching is still the same size
- -The blossom has an exposed vertex

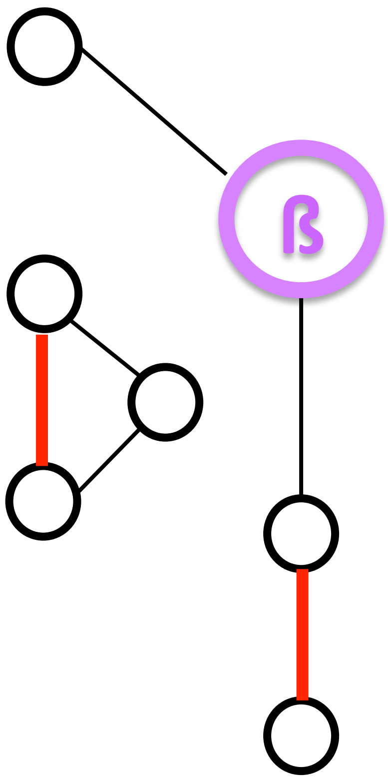
Theorem: M is maximum in G if and only if M-B is maximum in G-B



1- Flip the stem
- -The matching is still the same size
- -The blossom has an exposed vertex

2- Shrink the blossom to one single vertex
- -All vertices in B combine into ß
- -Edges into any vertex in B go into ß

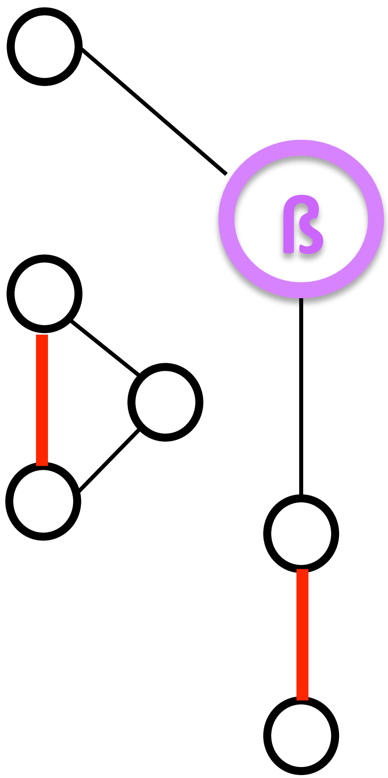
Theorem: M is maximum in G if and only if M-B is maximum in G-B



1- Flip the stem
- -The matching is still the same size
- -The blossom has an exposed vertex

2- Shrink the blossom to one single vertex
- -All vertices in B combine into ß
- -Edges into any vertex in B go into ß

<u>Theorem</u>: M is maximum in G if and only if M-B is maximum in G-B

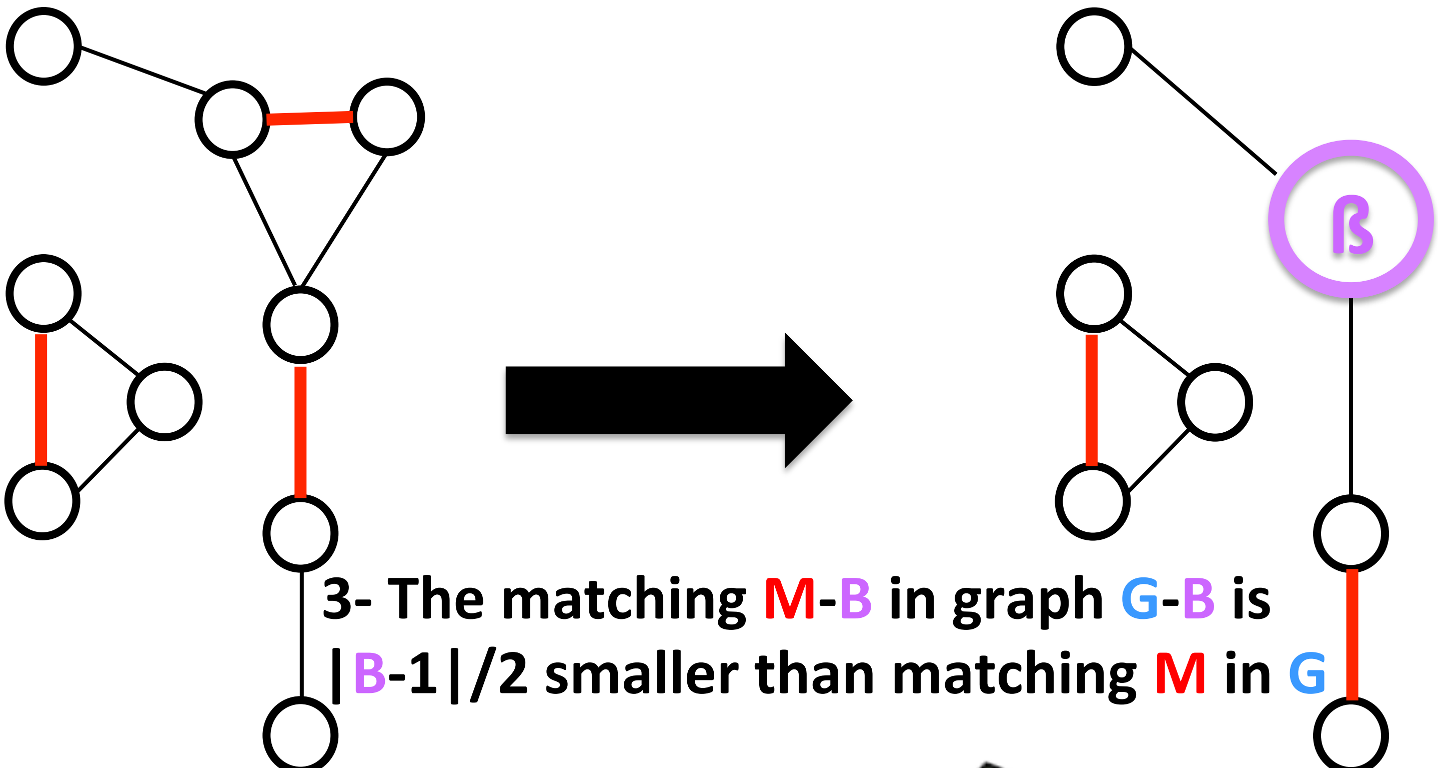


1- Flip the stem
- -The matching is still the same size
- -The blossom has an exposed vertex

2- Shrink the blossom to one single vertex
- -All vertices in B combine into ß
- -Edges into any vertex in B go into ß

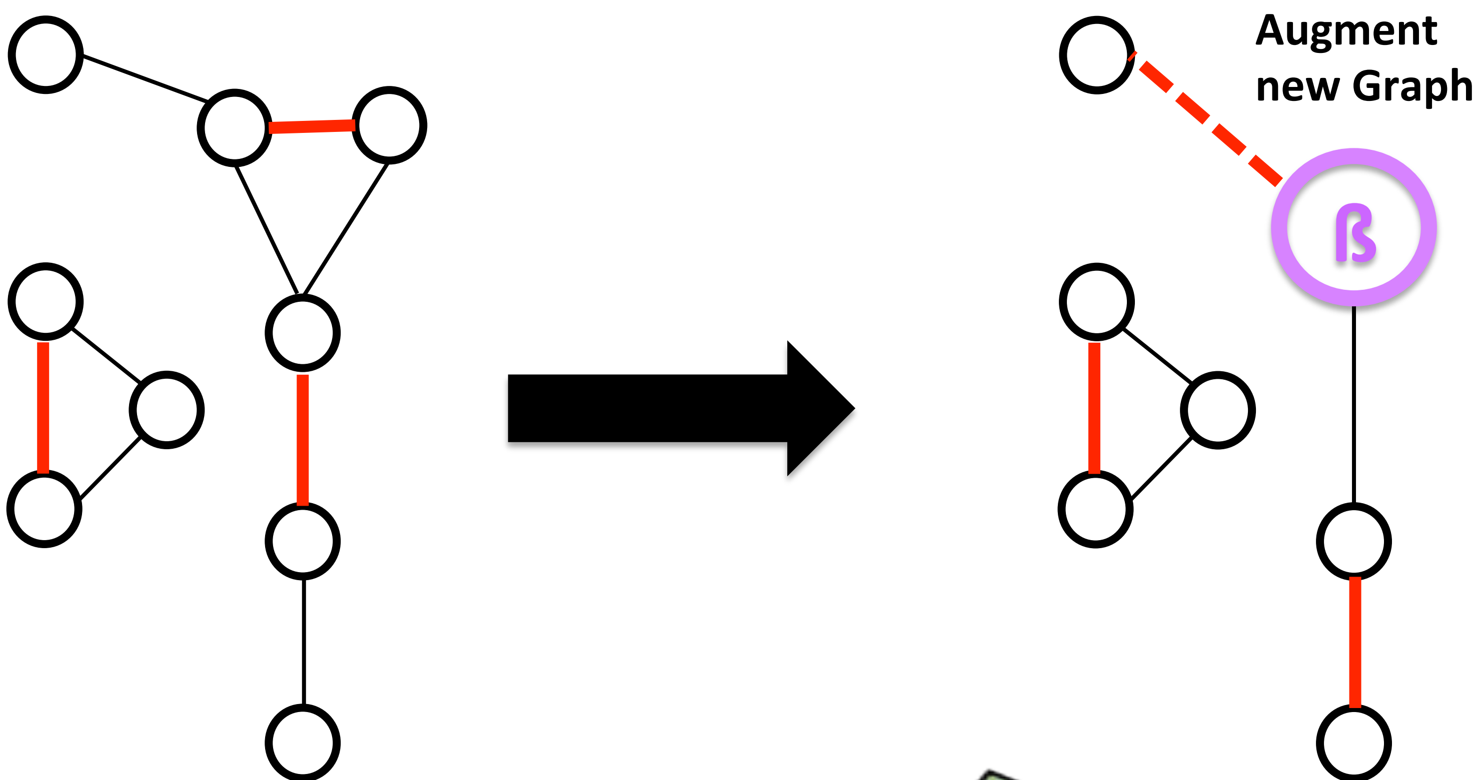
3- The matching M-B in graph G-B is |B-1|/2 smaller than matching M in G

Theorem: M is maximum in G if and only if M-B is maximum in G-B

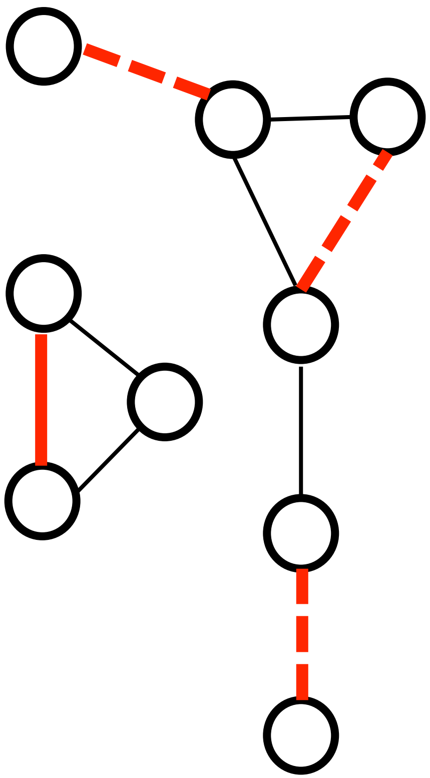


3- The matching M-B in graph G-B is |B-1|/2 smaller than matching M in G

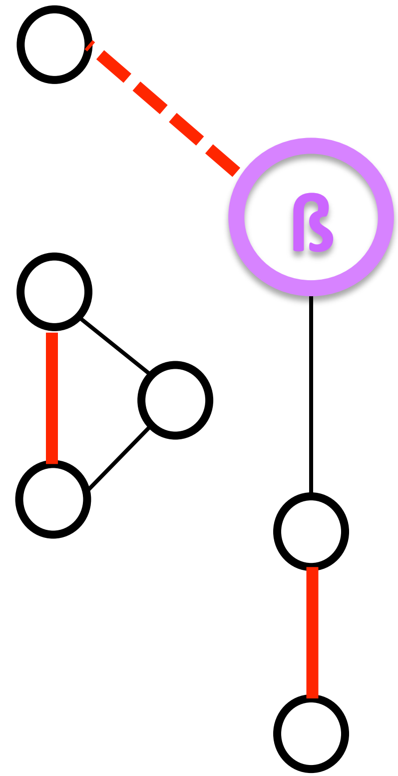
Increasing a Matching from a Flower



Increasing a Matching from a Flower



Add back in the blossom with extra edges from the new matching



Revised Algorithm

1 – Start with any matching **M** (let’s say M = {})

2 – Find a **flower**, **augmenting path** or **neither**:

- 3 – If **neither**: We’re done!
- 4 – If **augmenting path**: augment to **M’ = M Δ P**
- 5 – If **flower**: (recursively…)
  - a. Flip the stem
  - b. Shrink the blossom to a single vertex
  - c. Increase M or decide that it is Maximum

Revised Algorithm

1 – Start with any matching **M** (let's say M = {})

2 – Find a flower, augmenting path or neither:

3 – If **neither**: We're done!

4 – If **augmenting path**: augment to **M' = M Δ P**

5 – If **flower**: (recursively…)

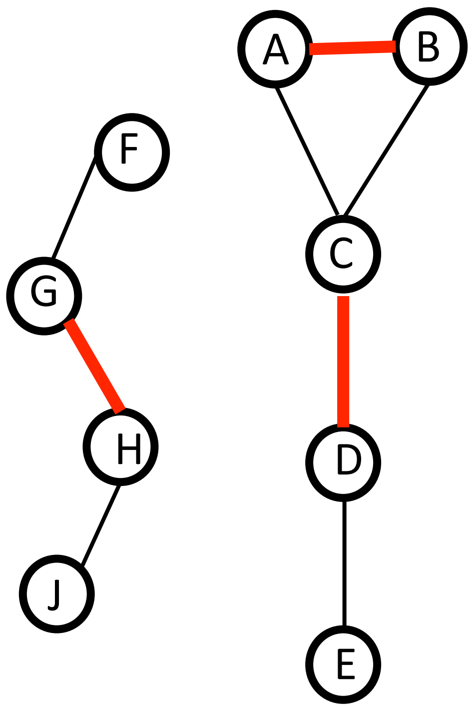
a. Flip the stem

b. Shrink the blossom to a single vertex

c. Increase M or decide that it is Maximum

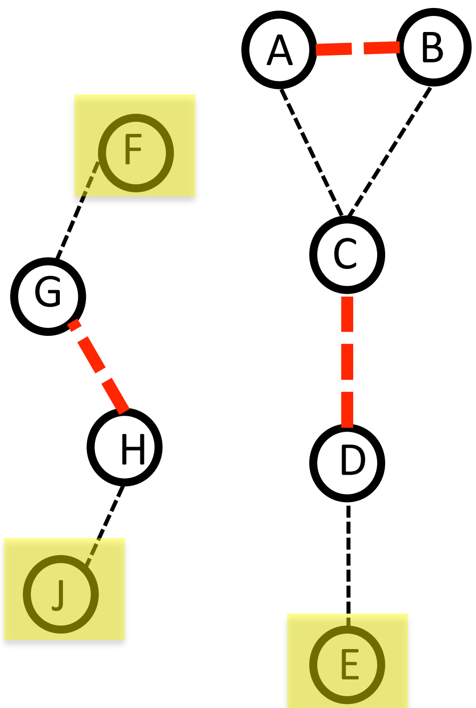
HOW???

Finding Flowers and Augmenting Paths

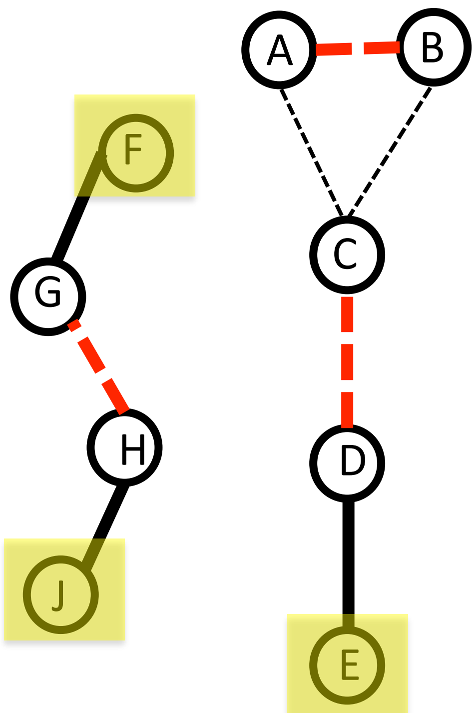


Creating Alternating Forests

1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one

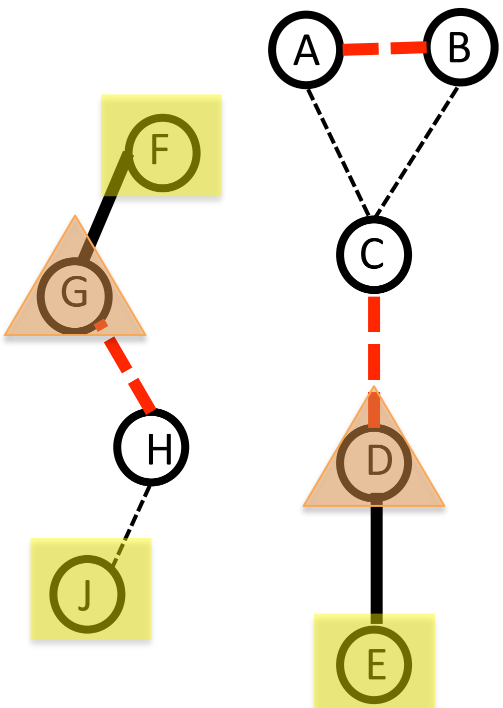


Creating Alternating Forests



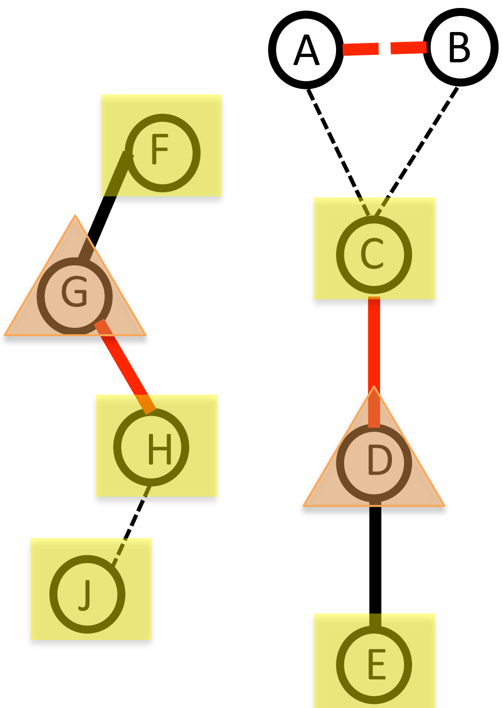
1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (*u*, *v*) from *u* in the forest to *v*

Creating Alternating Forests



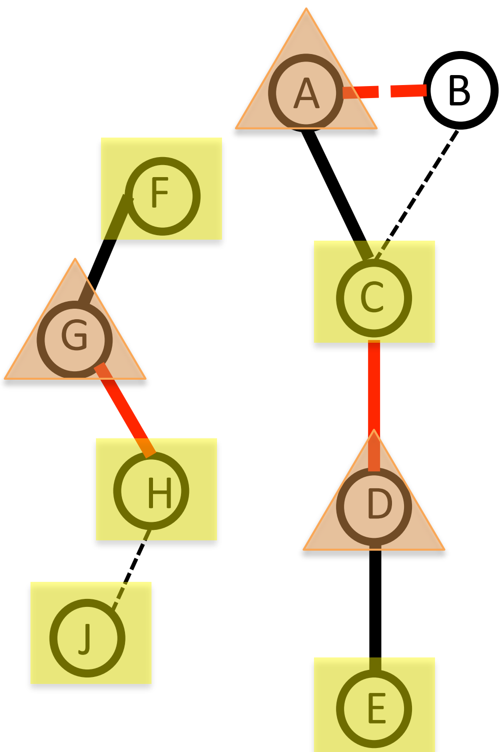
1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (u, v) from u in the forest to v
3. If an edge (*u*, *v*) has *v* unlabelled, label it TRIANGLE. Label its "mate" (across an edge in the matching) as a SQUARE

Creating Alternating Forests



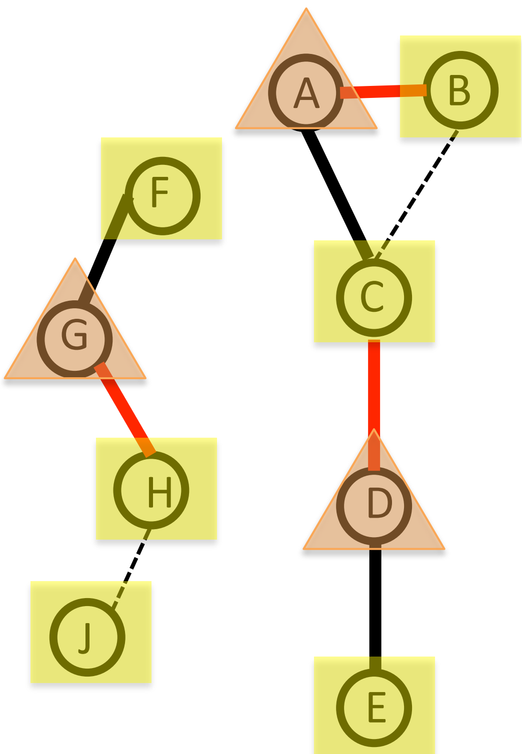
1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (u, v) from u in the forest to v
3. If an edge (u, v) has v unlabelled, label it TRIANGLE. **Label its "mate" (across an edge in the matching) as a SQUARE**

Creating Alternating Forests



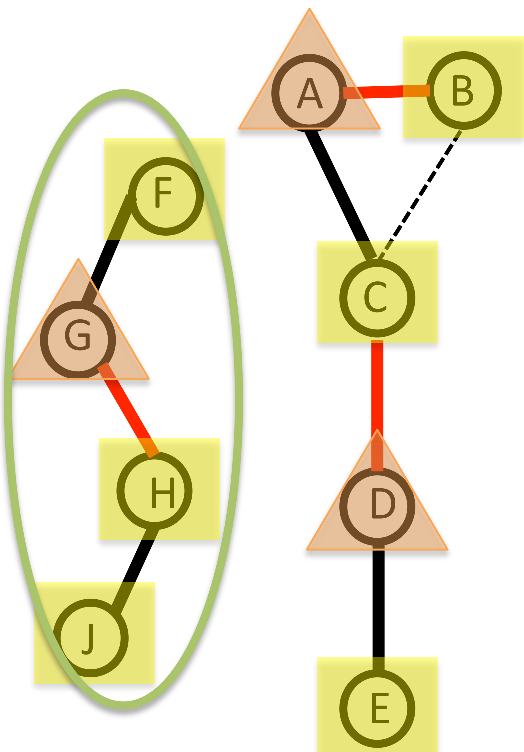
1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (*u, v*) from *u* in the forest to *v*
3. If an edge (*u, v*) has *v* unlabelled, label it TRIANGLE. Label its "mate" (across an edge in the matching) as a SQUARE

Creating Alternating Forests



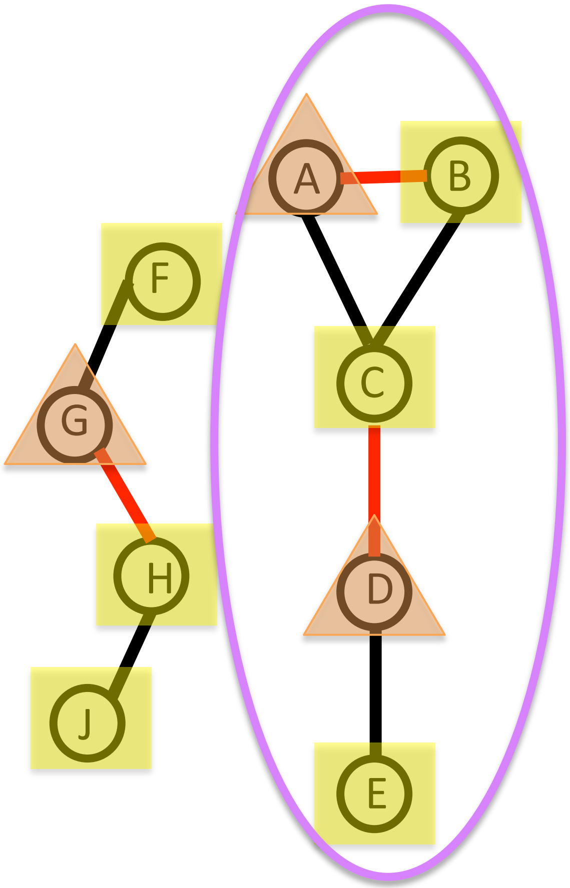
1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (*u, v*) from *u* in the forest to *v*
3. If an edge (*u, v*) has *v* unlabelled, label it TRIANGLE. **Label its "mate" (across an edge in the matching) as a SQUARE**

Creating Alternating Forests



1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (*u, v*) from *u* in the forest to *v*
3. If an edge (*u, v*) has *v* unlabelled, label it TRIANGLE. Label its "mate" (across an edge in the matching) as a SQUARE
4. If an edge (*u, v*) already has *v* labelled SQUARE and *v* belongs to a different alternating tree, then we have an augmenting path

Creating Alternating Forests



1. Label all exposed vertices as SQUARE, start a new tree in our alternating forest for each one
2. Add edges (*u, v*) from *u* in the forest to *v*
3. If an edge (*u, v*) has *v* unlabelled, label it TRIANGLE. Label its "mate" (across an edge in the matching) as a SQUARE
4. If an edge (*u, v*) already has *v* labelled SQUARE and *v* belongs to a different alternating tree, then we have an augmenting path
5. If an edge (*u, v*) already has *v* labelled SQUARE and *v* belongs to the same alternating tree, then we have found a flower

Questions?