



# Chapter 5

# Graph Algorithms

Matching

**Algorithm Theory**

**WS 2014/15**

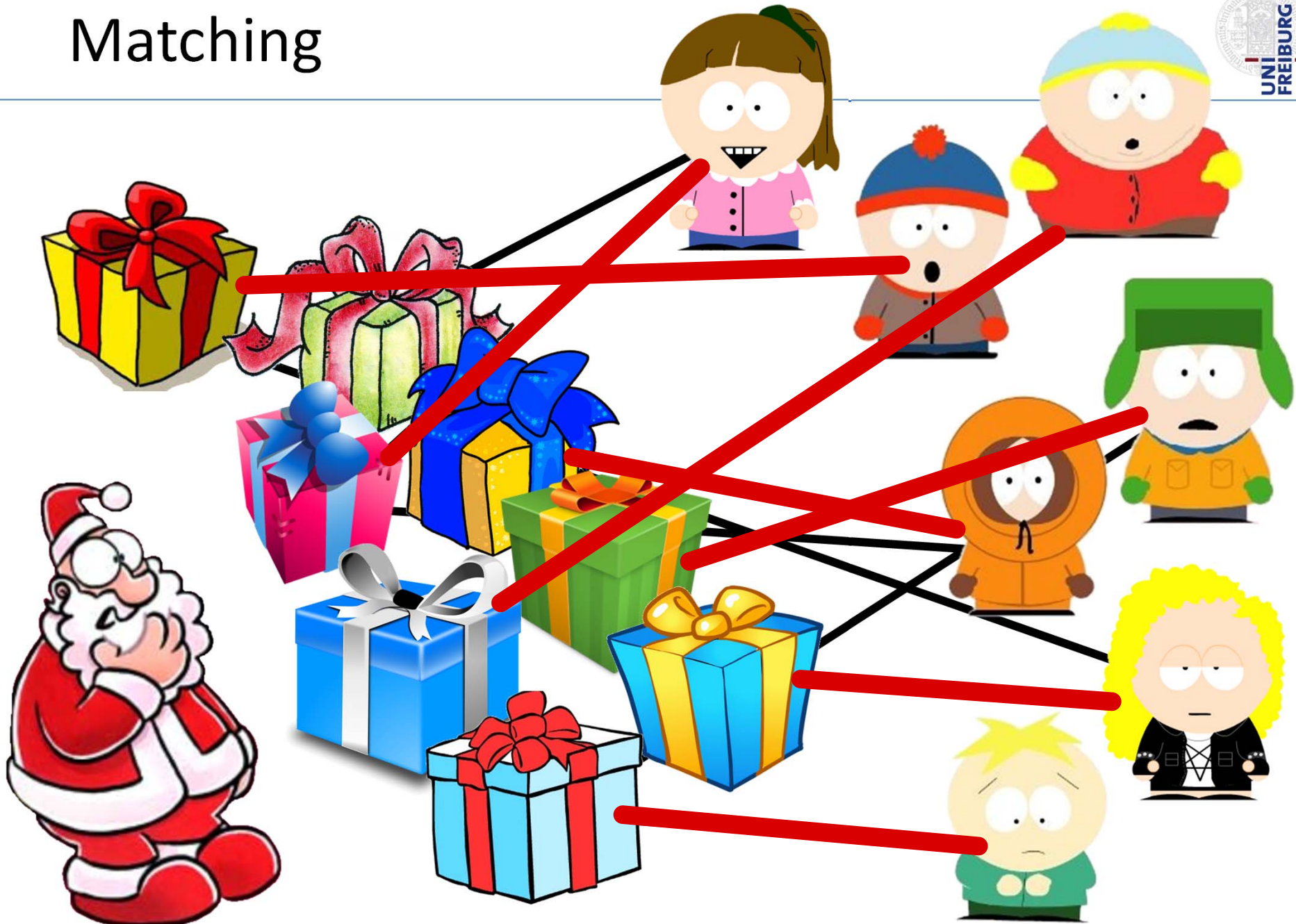
**Fabian Kuhn**

Monday, Dec 22

Exercises

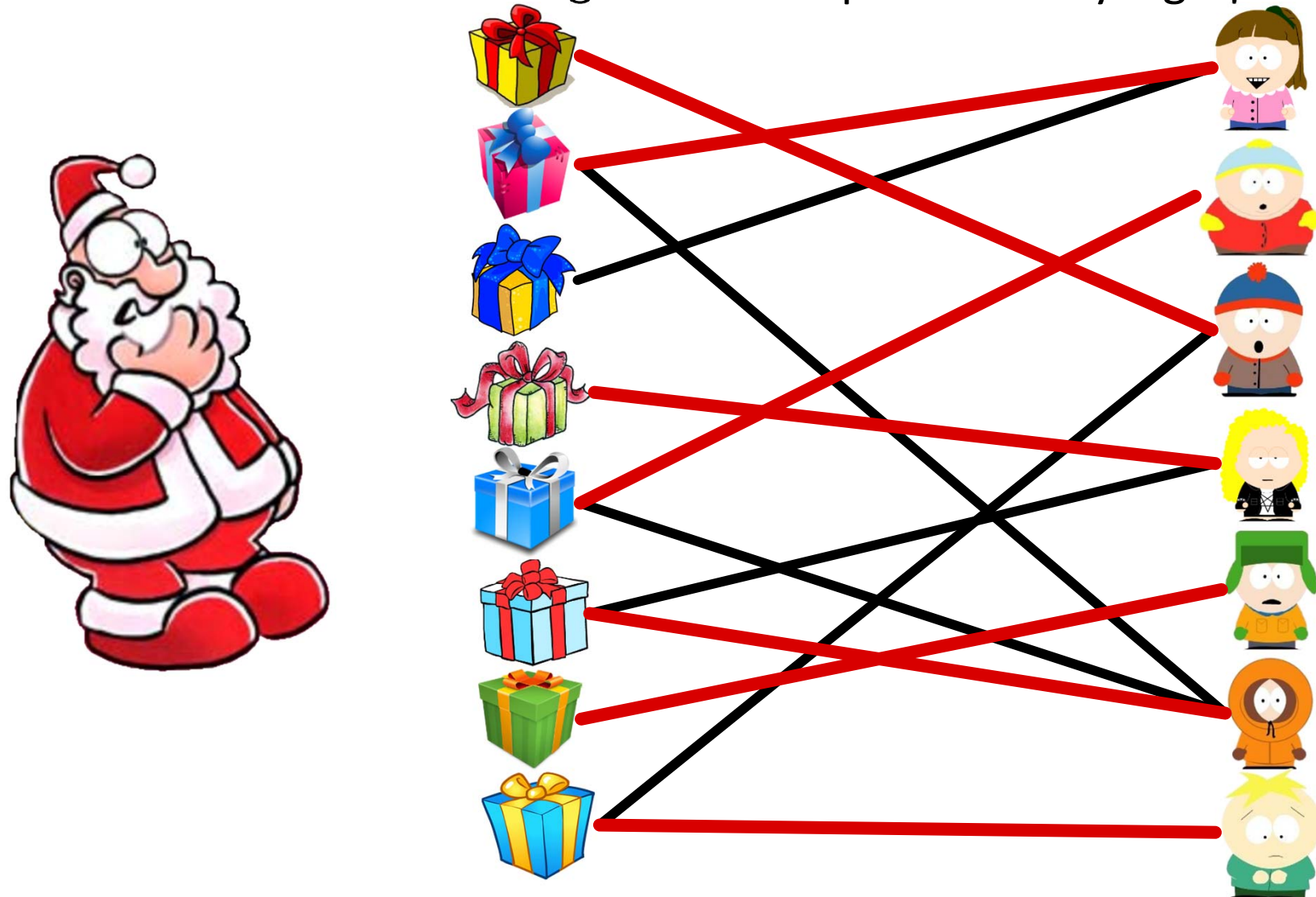
14<sup>15</sup> - 16<sup>00</sup>

# Matching



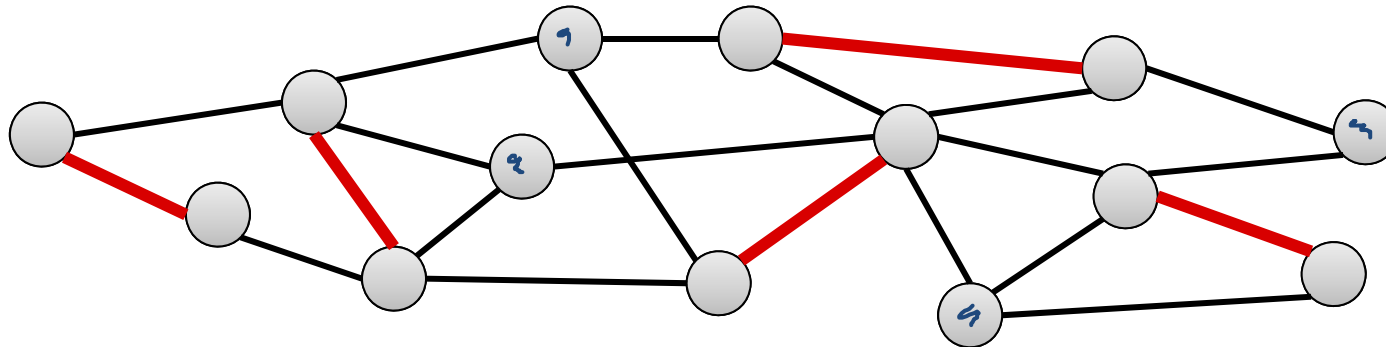
# Gifts-Children Graph

- Which child likes which gift can be represented by a graph



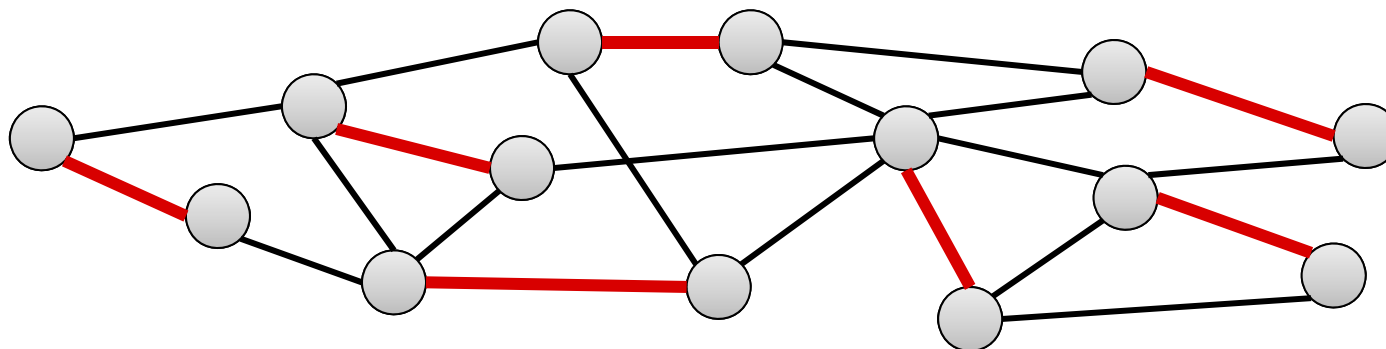
# Matching

**Matching:** Set of pairwise non-incident edges



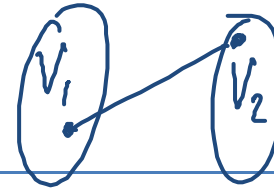
**Maximal Matching:** A matching s.t. no more edges can be added

**Maximum Matching:** A matching of maximum possible size



**Perfect Matching:** Matching of size  $\frac{n}{2}$  (every node is matched)

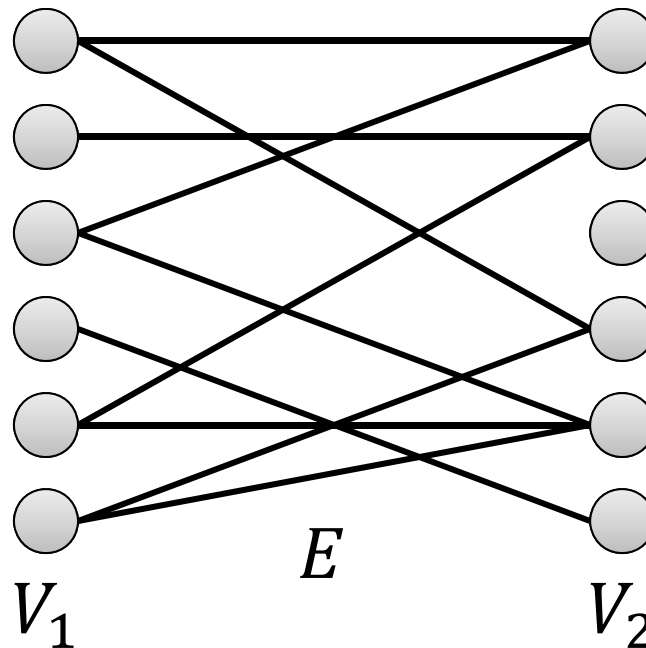
# Bipartite Graph



**Definition:** A graph  $G = (V, E)$  is called bipartite iff its node set can be partitioned into two parts  $V = \underline{V_1} \cup \underline{V_2}$  such that for each edge  $\{u, v\} \in E$ ,

$$|\{u, v\} \cap V_1| = 1.$$

- Thus, edges are only between the two parts



# Santa's Problem

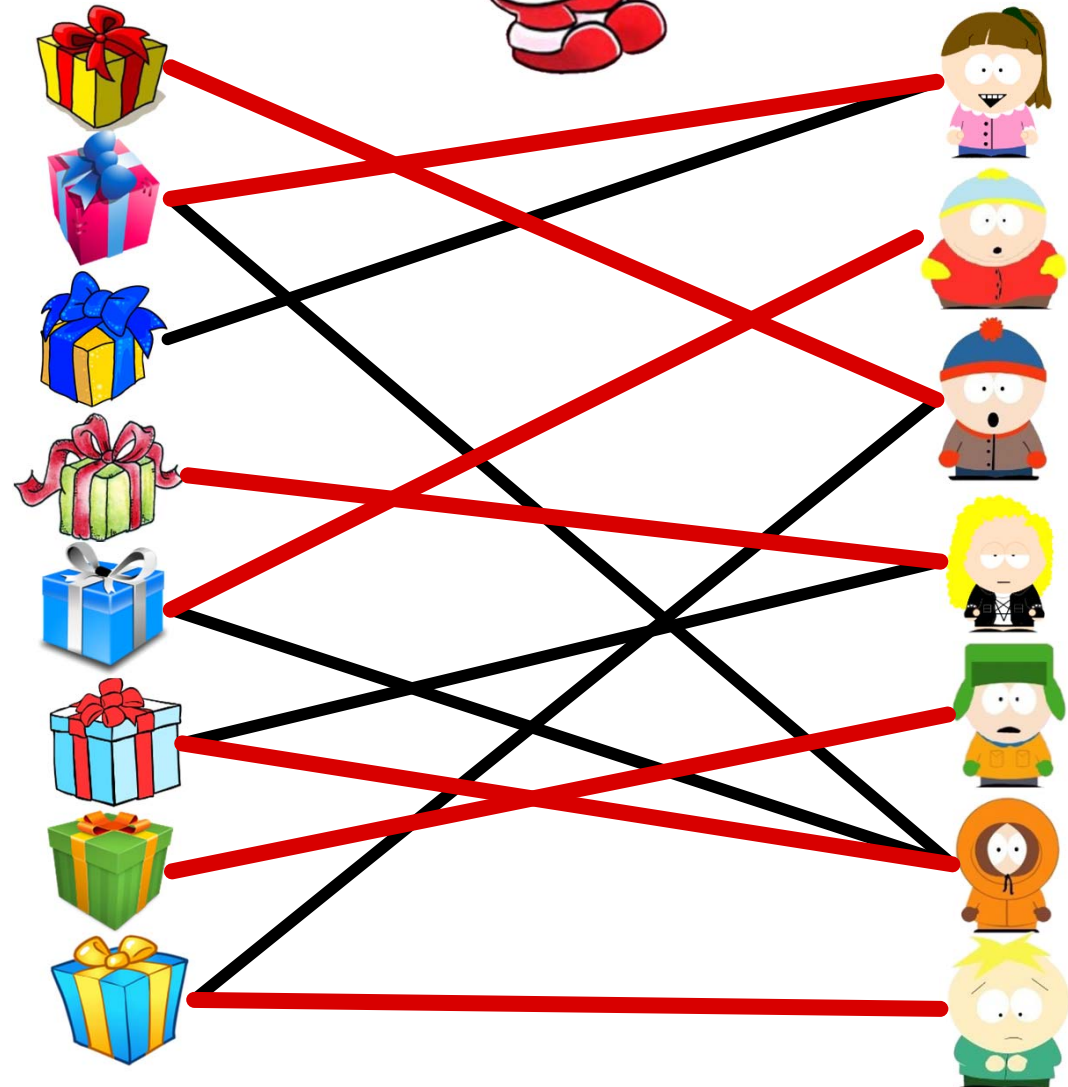


## Maximum Matching in Bipartite Graphs:

Every child can get a gift iff there is a matching of size  $\#$ children

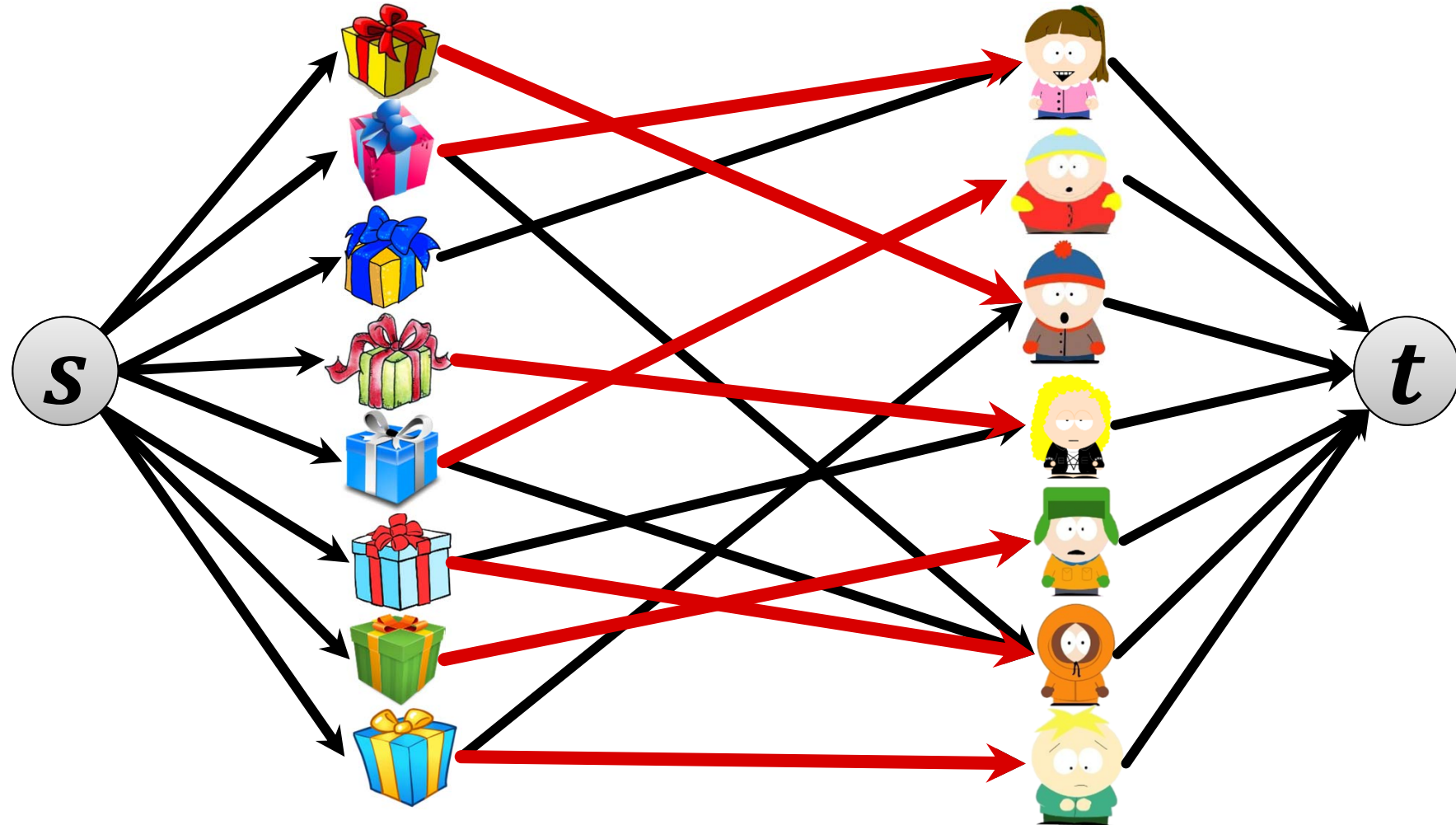
Clearly, every matching is at most as big

If  $\#$ children =  $\#$ gifts, there is a solution iff there is a perfect matching



# Reducing to Maximum Flow

- Like edge-disjoint paths...



**all capacities are 1**

# Reducing to Maximum Flow

**Theorem:** Every integer solution to the max flow problem on the constructed graph induces a maximum bipartite matching of  $G$ .

**Proof:**

1. An integer flow  $f$  of value  $|f|$  induces a matching of size  $|f|$ 
  - Left nodes (gifts) have incoming capacity 1
  - Right nodes (children) have outgoing capacity 1
  - Left and right nodes are incident to  $\leq 1$  edge  $e$  of  $G$  with  $f(e) = 1$
  
2. A matching of size  $k$  implies a flow  $f$  of value  $|f| = k$ 
  - For each edge  $\{u, v\}$  of the matching:
 
$$f((s, u)) = f((u, v)) = f((v, t)) = 1$$
  - All other flow values are 0



# Running Time of Max. Bipartite Matching



**Theorem:** A maximum matching of a bipartite graph can be computed in time  $O(m \cdot n)$ .

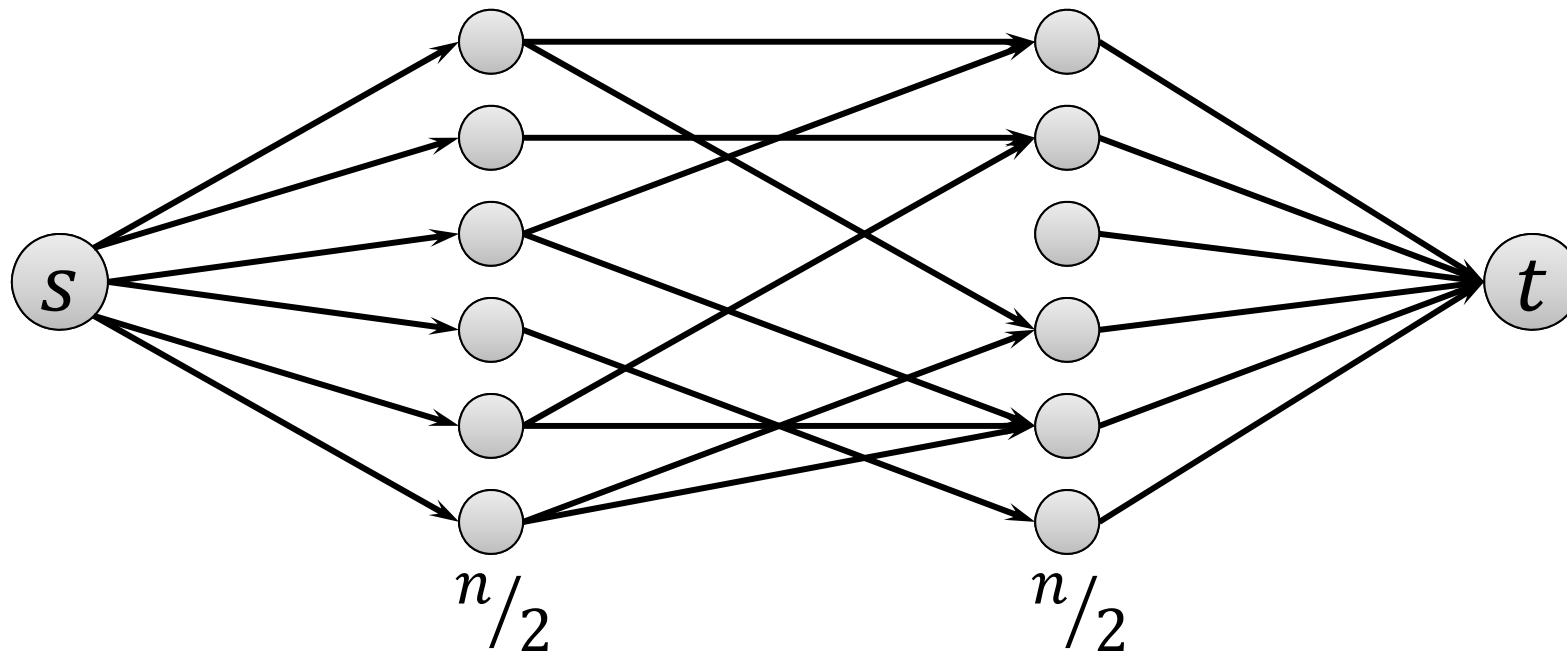
use Ford Fulkerson:

1 iteration :  $O(m)$

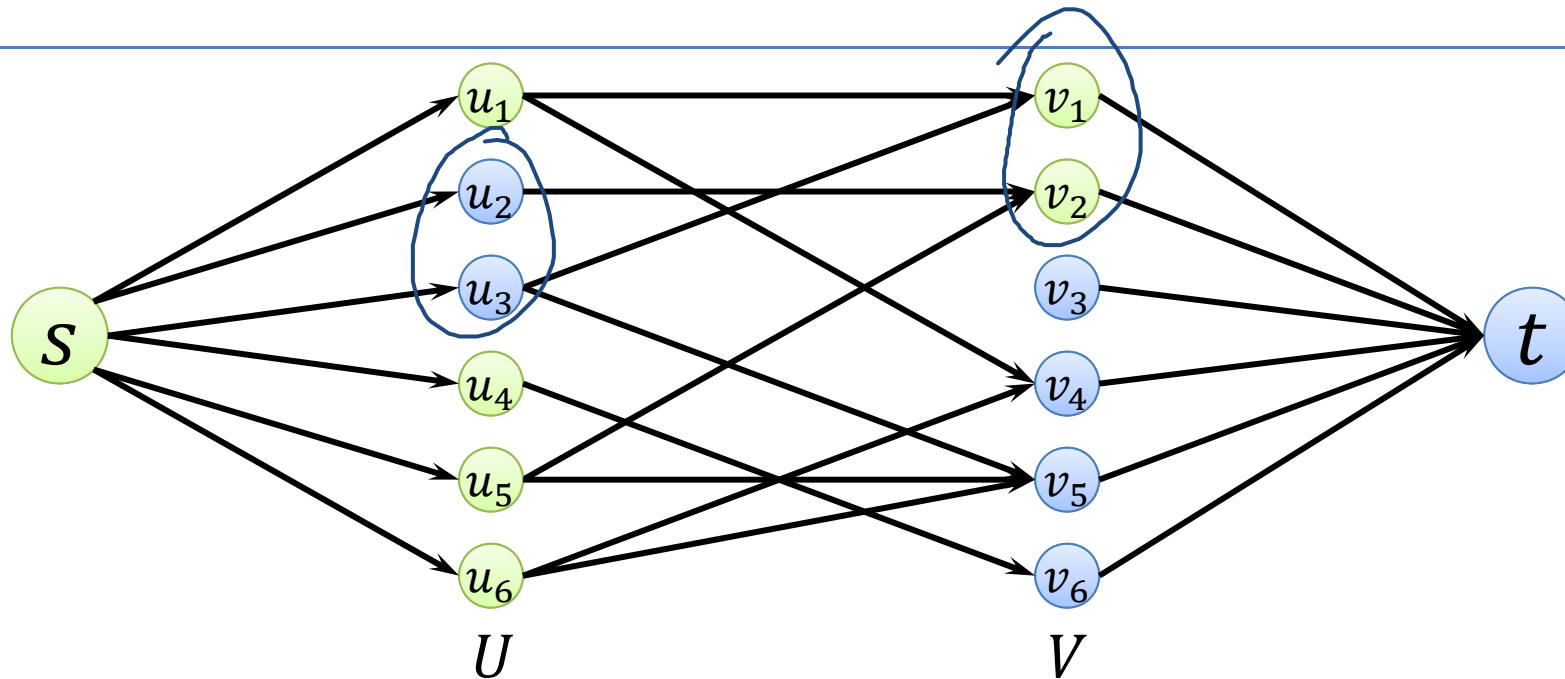
# iter: size of maximum matching  $\leq \frac{n}{2}$

# Perfect Matching?

- There can only be a perfect matching if both sides of the partition have size  $n/2$ .
- There is no perfect matching, iff there is an  $s-t$  cut of size  $< n/2$  in the flow network.



# $s$ - $t$ Cuts



Partition  $(A, B)$  of node set such that  $s \in A$  and  $t \in B$

- If  $v_i \in A$ : edge  $(v_i, t)$  is in cut  $(A, B)$
- If  $u_i \in B$ : edge  $(s, u_i)$  is in cut  $(A, B)$
- Otherwise (if  $u_i \in A, v_i \in B$ ), all edges from  $u_i$  to some  $v_j \in B$  are in cut  $(A, B)$

# Hall's Marriage Theorem

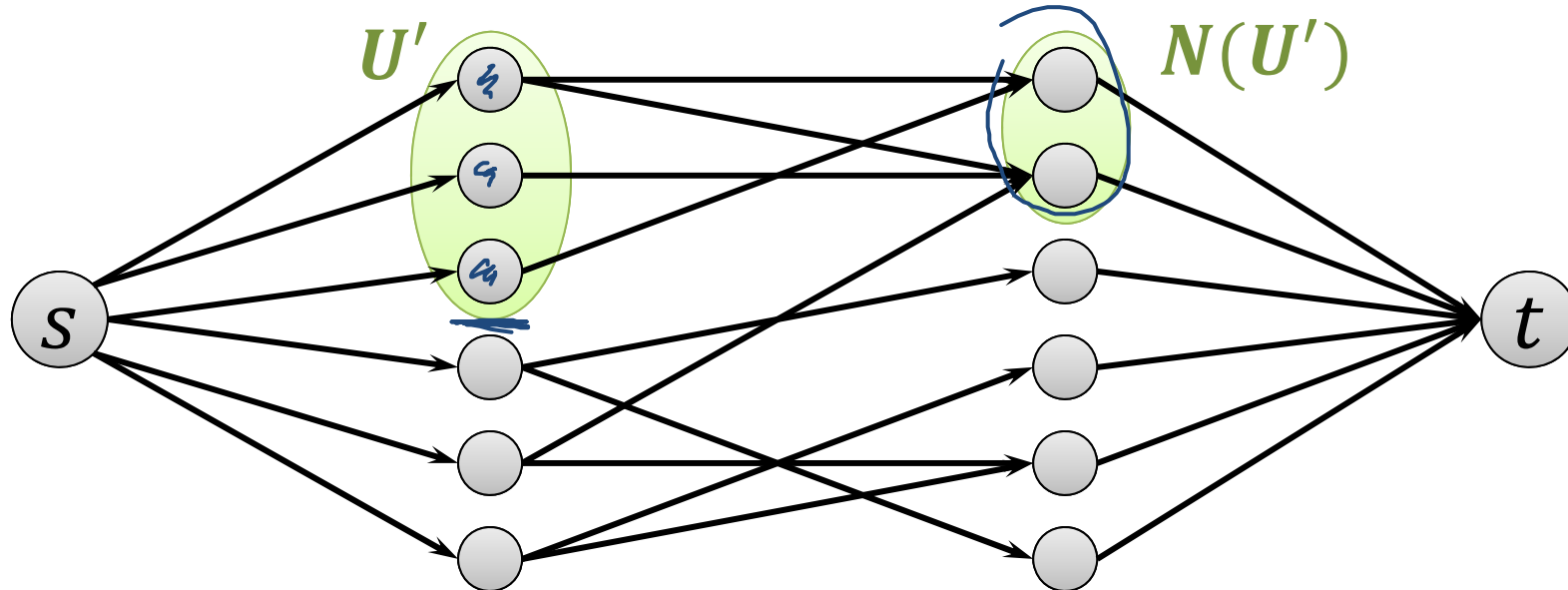
**Theorem:** A bipartite graph  $G = (U \cup V, E)$  for which  $|U| = |V|$  has a perfect matching if and only if

$$\forall U' \subseteq U: |N(U')| \geq |U'|,$$

where  $N(U') \subseteq V$  is the set of neighbors of nodes in  $U'$ .

**Proof:** No perfect matching  $\Leftrightarrow$  some  $s$ - $t$  cut has capacity  $< n/2$

1. Assume there is  $U'$  for which  $|N(U')| < |U'|$ :



# Hall's Marriage Theorem

*need to show*  
 $\exists u': |N(u')| < |u'|$



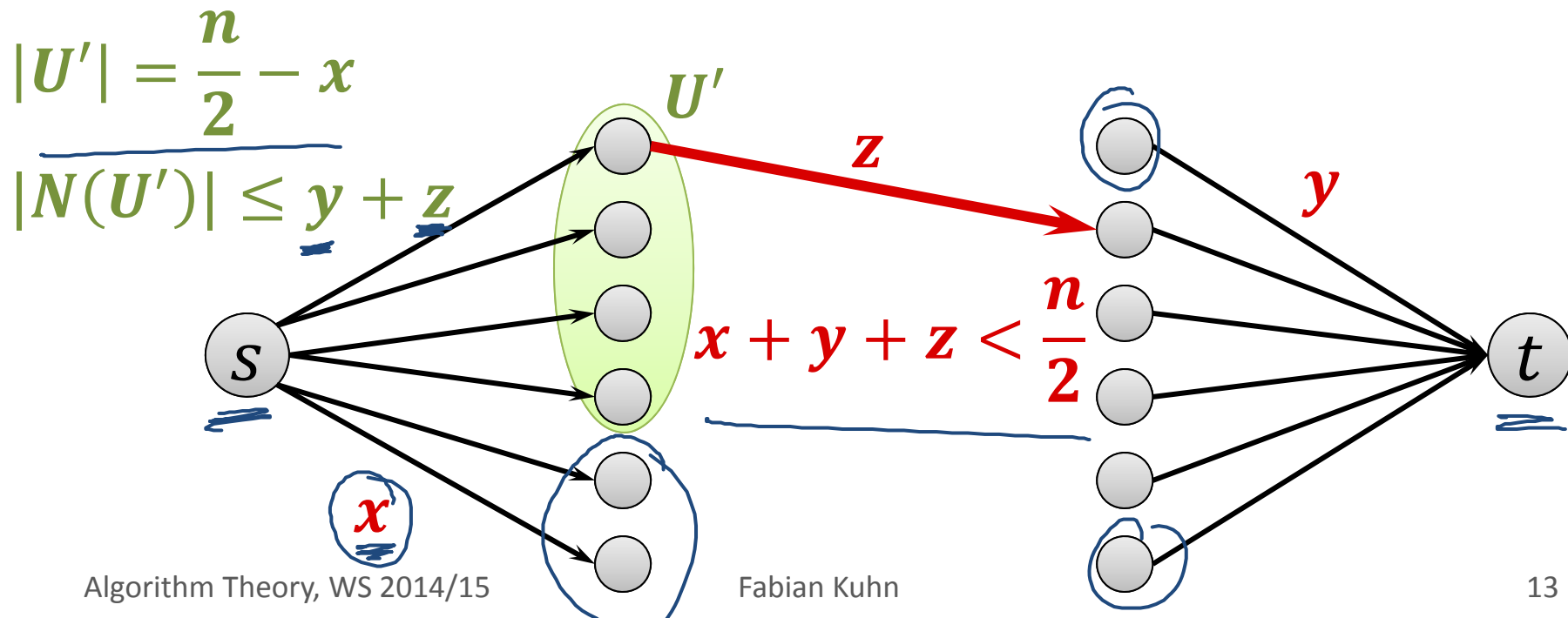
**Theorem:** A bipartite graph  $G = (U \cup V, E)$  for which  $|U| = |V|$  has a perfect matching if and only if

$$\forall U' \subseteq U: |N(U')| \geq |U'|,$$

where  $N(U') \subseteq V$  is the set of neighbors of nodes in  $U'$ .

**Proof:** No perfect matching  $\Leftrightarrow$  some  $s$ - $t$  cut has capacity  $< n/2$

2. Assume that there is a cut  $(A, B)$  of capacity  $< n/2$



# Hall's Marriage Theorem

**Theorem:** A bipartite graph  $G = (U \cup V, E)$  for which  $|U| = |V|$  has a perfect matching if and only if

$$\forall U' \subseteq U: |N(U')| \geq |U'|,$$

where  $N(U') \subseteq V$  is the set of neighbors of nodes in  $U'$ .

**Proof:** No perfect matching  $\Leftrightarrow$  some  $s$ - $t$  cut has capacity  $< n$

2. Assume that there is a cut  $(A, B)$  of capacity  $< n$

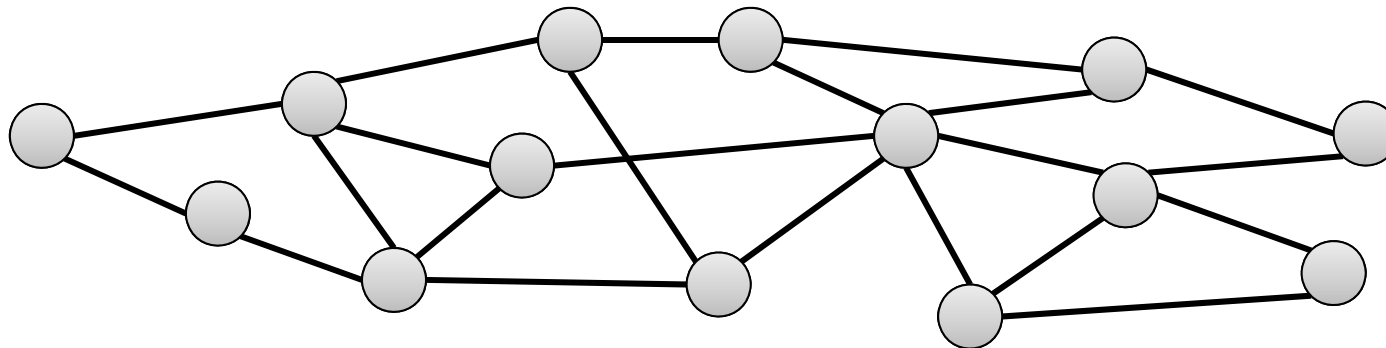
$$\begin{array}{l}
 |U'| = \frac{n}{2} - x \\
 \hline
 |N(U')| \leq y + z \\
 \hline
 x + y + z < \frac{n}{2}
 \end{array}
 \quad \Rightarrow \quad
 \underbrace{y + z}_{\geq |N(U')|} < \frac{n}{2} - x = |U'|$$

$\underbrace{|N(U')| < |U'|}$

# What About General Graphs

- Can we efficiently compute a maximum matching if  $G$  is not bipartite?
- How good is a **maximal matching**?
  - A matching that cannot be extended...

- **Vertex Cover:** set  $S \subseteq V$  of nodes such that
 
$$\forall \{u, v\} \in E, \quad \{u, v\} \cap S \neq \emptyset.$$

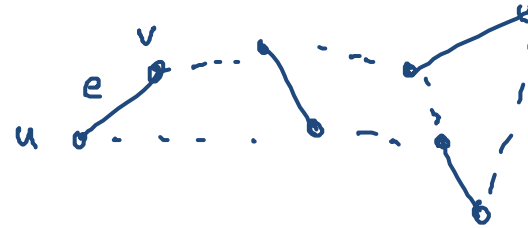


- A vertex cover covers all edges by incident nodes

# Vertex Cover vs Matching

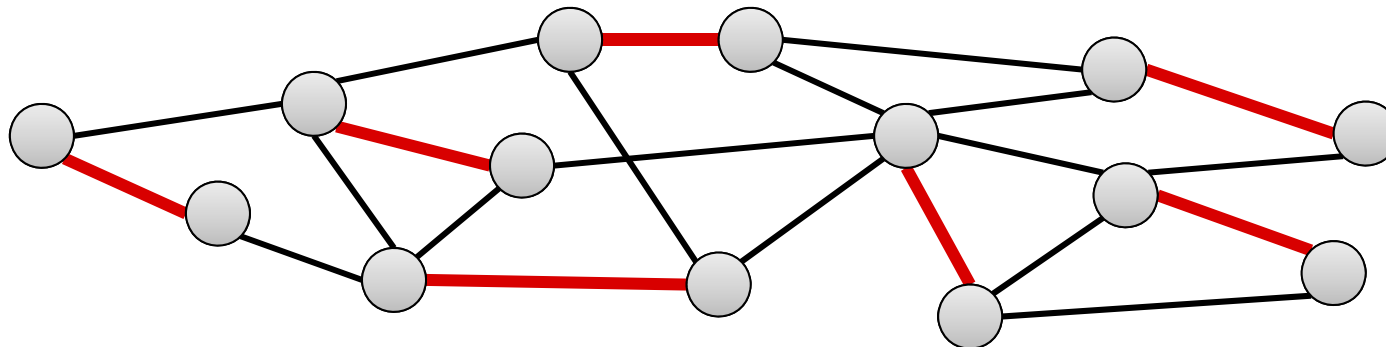
Consider a matching  $M$  and a vertex cover  $S$

**Claim:**  $|M| \leq |S|$



**Proof:**

- At least one node of every edge  $\{u, v\} \in M$  is in  $S$
- Needs to be a different node for different edges from  $M$





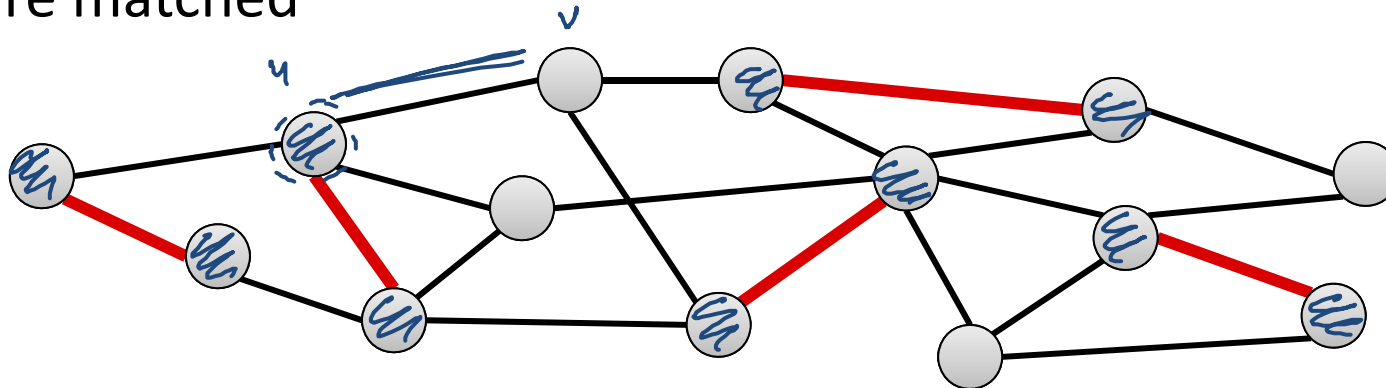
# Vertex Cover vs Matching

Consider a matching  $M$  and a vertex cover  $S$

**Claim:** If  $M$  is maximal and  $S$  is minimum,  $|S| \leq 2|M|$

**Proof:**

- $M$  is maximal: for every edge  $\{u, v\} \in E$ , either  $u$  or  $v$  (or both) are matched



- Every edge  $e \in E$  is “covered” by at least one matching edge
- Thus, the set of the nodes of all matching edges gives a vertex cover  $S$  of size  $|S| = 2|M|$ .

# Maximal Matching Approximation

**Theorem:** For any maximal matching  $M$  and any maximum matching  $M^*$ , it holds that

$$|M| \geq \frac{|M^*|}{2}$$

2-approximation

**Proof:**

$S^*$ : opt. vert. cover

$$\underline{|M^*|} \stackrel{\text{Lem. 1}}{\leq} |S^*| \stackrel{\text{Lem. 2}}{\leq} \underline{2|M|}$$

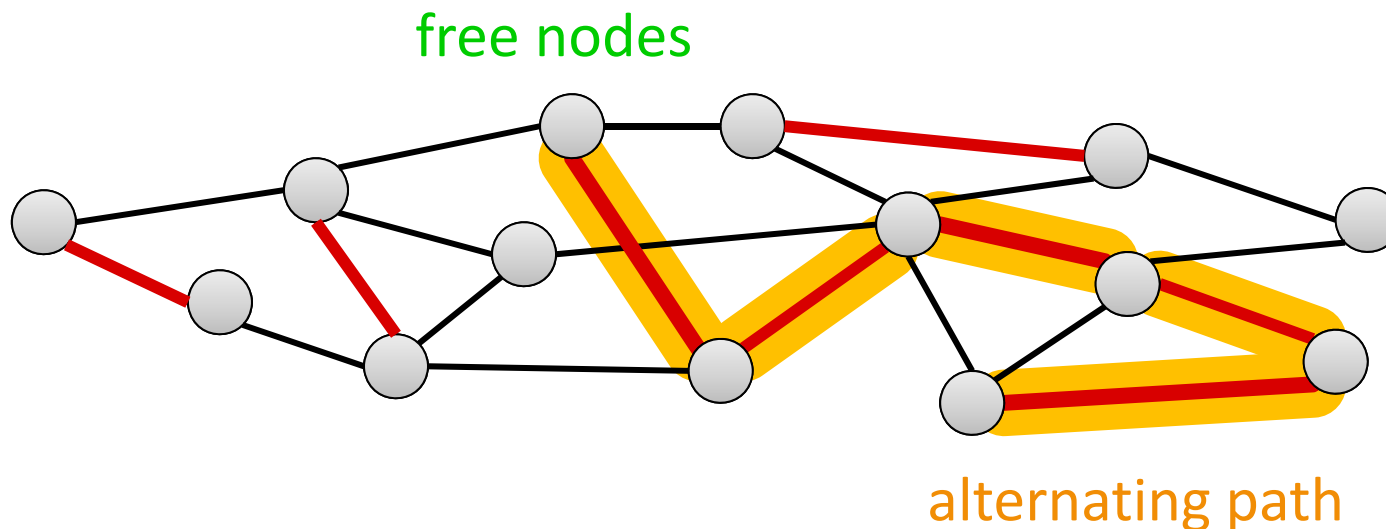
**Theorem:** The set of all matched nodes of a maximal matching  $M$  is a vertex cover of size at most twice the size of a min. vertex cover.

# Augmenting Paths

Consider a matching  $M$  of a graph  $G = (V, E)$ :

- A **node**  $v \in V$  is called **free** iff it is **not matched**

**Augmenting Path:** A (odd-length) path that starts and ends at a free node and visits edges in  $E \setminus M$  and edges in  $M$  alternately.



- Matching  $M$  can be improved using an augmenting path by switching the role of each edge along the path

# Augmenting Paths

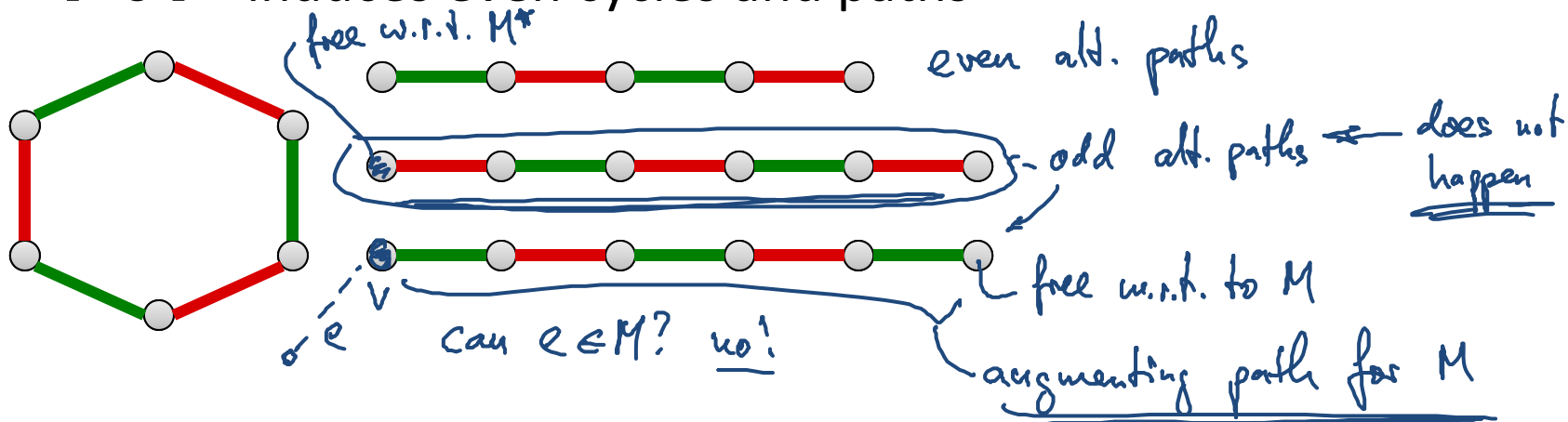
**Theorem:** A matching  $M$  of  $G = (V, E)$  is maximum if and only if there is no augmenting path.

**Proof:**

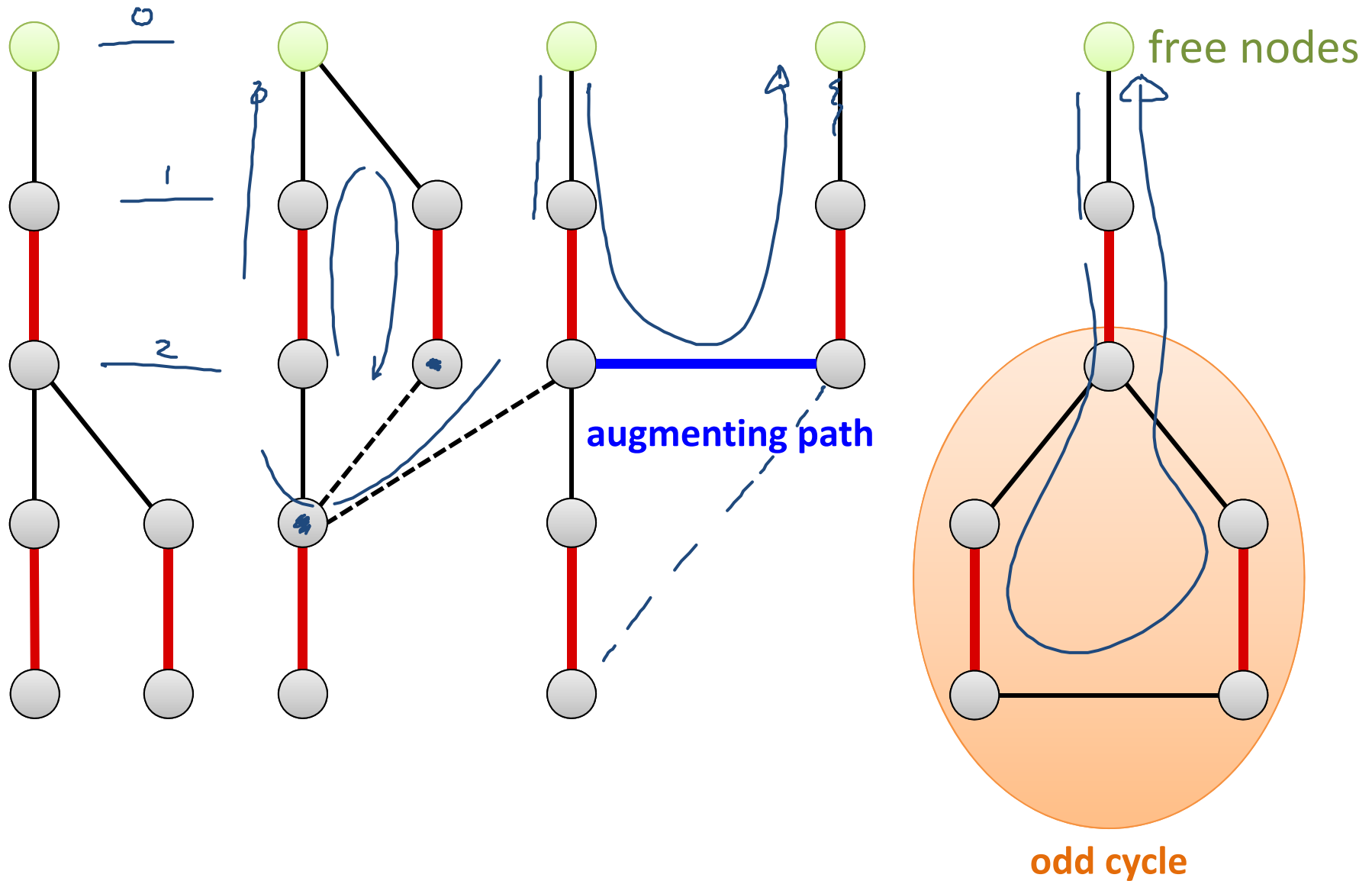
- Consider non-max. matching  $M$  and max. matching  $M^*$  and define

$$F := M \setminus M^*, \quad F^* := M^* \setminus M$$

- Note that  $F \cap F^* = \emptyset$  and  $|F| < |F^*|$  (because  $|M| < |M^*|$ )
- Each node  $v \in V$  is incident to at most one edge in both  $F$  and  $F^*$
- $F \cup F^*$  induces even cycles and paths

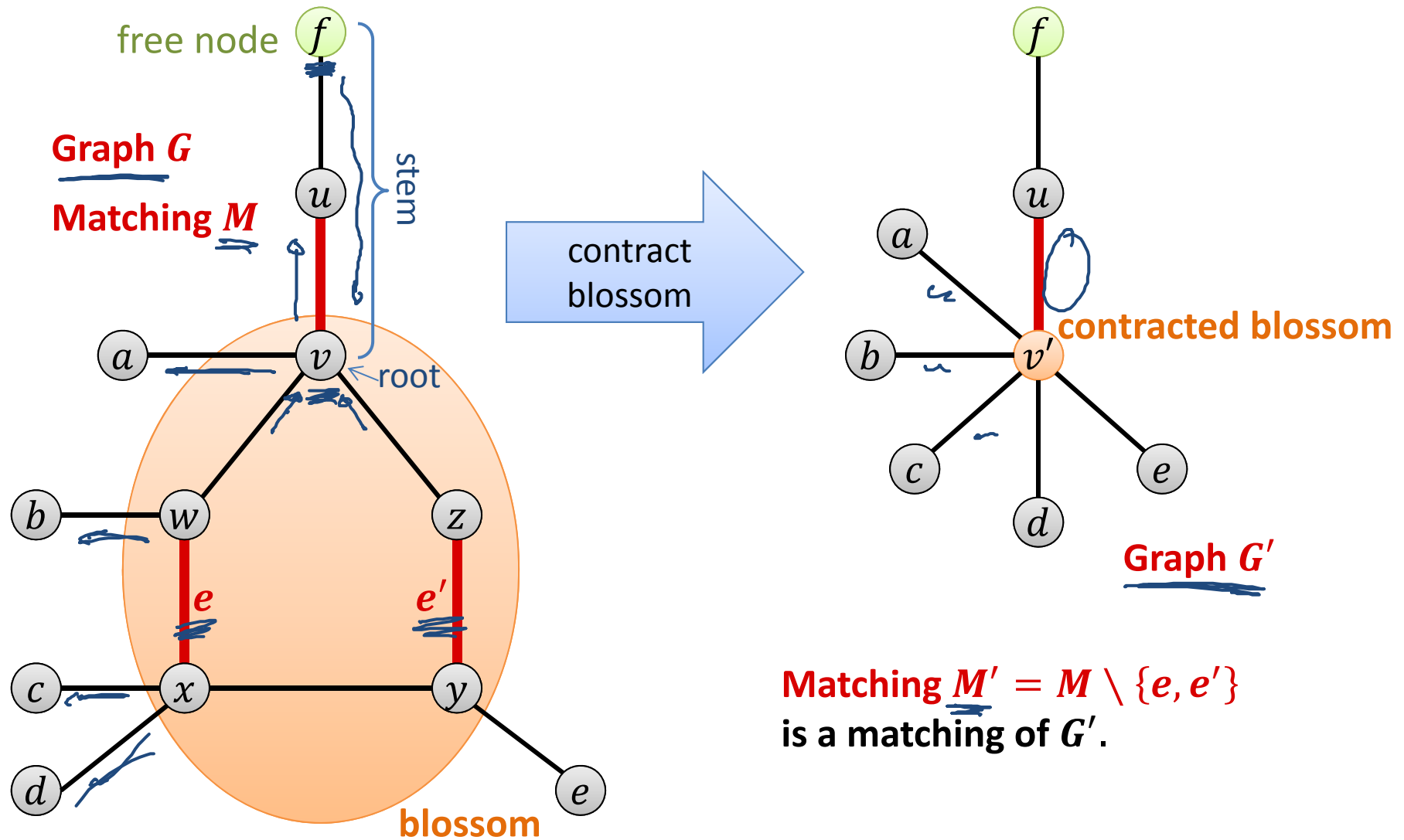


# Finding Augmenting Paths



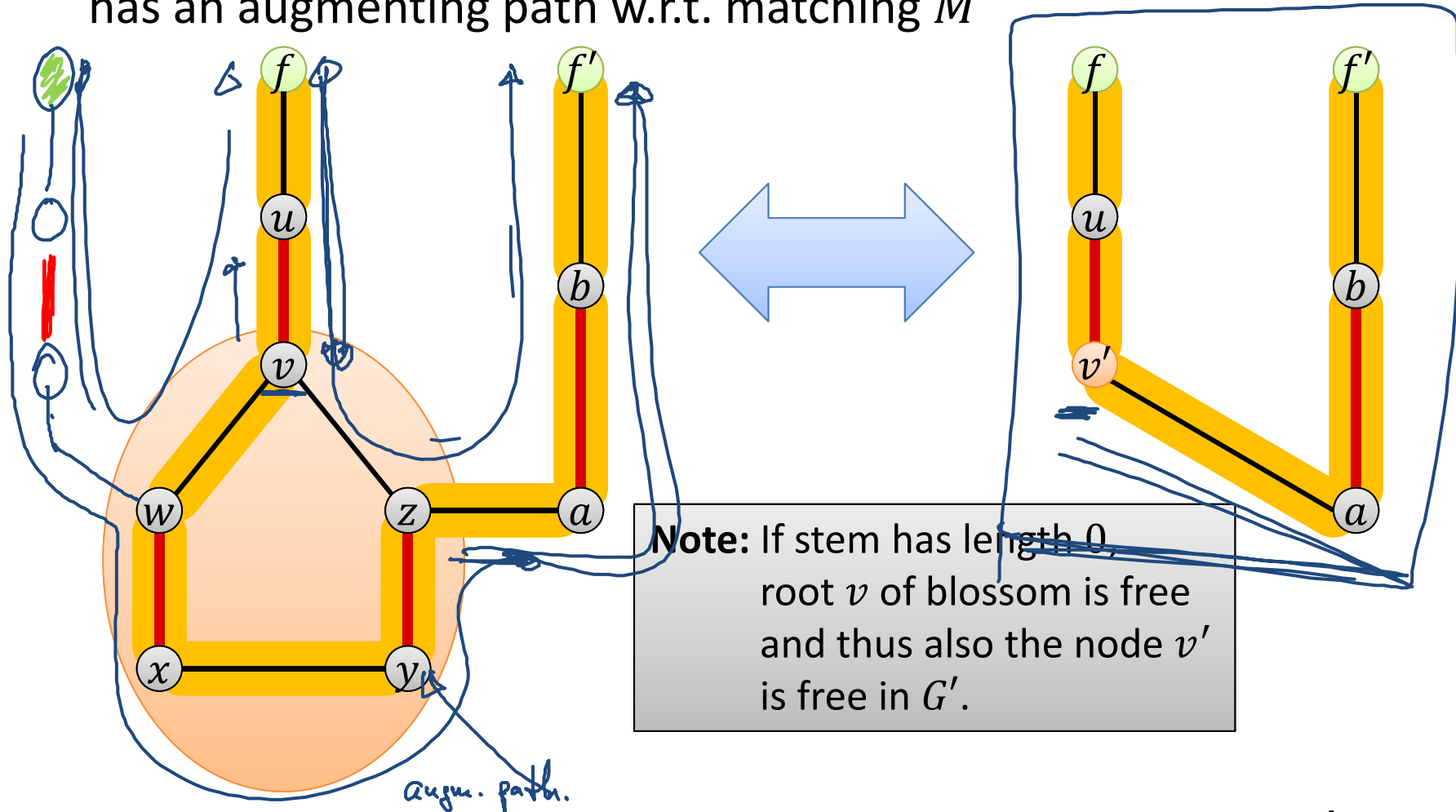
# Blossoms

- If we find an odd cycle...



# Contracting Blossoms

**Lemma:** Graph  $G$  has an augmenting path w.r.t. matching  $M$  iff  $G'$  has an augmenting path w.r.t. matching  $M'$



**Note:** If stem has length 0, root  $v$  of blossom is free and thus also the node  $v'$  is free in  $G'$ .

**Also:** The matching  $M$  can be computed efficiently from  $M'$ .

# Edmond's Blossom Algorithm



## Algorithm Sketch:



1. Build a tree for each free node
2. Starting from an explored node  $u$  at even distance from a free node  $f$  in the tree of  $f$ , explore some unexplored edge  $\{u, v\}$ :

1. If  $v$  is an unexplored node,  $v$  is matched to some neighbor  $w$ :  
add  $w$  to the tree ( $w$  is now explored)

2. If  $v$  is explored and in the same tree:  
at odd distance from root  $\rightarrow$  ignore and move on  
at even distance from root  $\rightarrow$  **blossom found**

3. If  $v$  is explored and in another tree  
at odd distance from root  $\rightarrow$  ignore and move on  
at even distance from root  $\rightarrow$  **augmenting path found**



# Running Time

---

**Finding a Blossom:** Repeat on smaller graph

**Finding an Augmenting Path:** Improve matching

**Theorem:** The algorithm can be implemented in time  $O(mn^2)$ .

$$\# \text{ augm. paths (matching improvements)} \leq \frac{n}{2}$$

$$| \text{ exploration} : O(m)$$

$$\# \text{ explorations per augm. path} : \leq \frac{n}{2}$$

# Matching Algorithms

## We have seen:

- $O(mn)$  time alg. to compute a max. matching in *bipartite graphs*
- $O(mn^2)$  time alg. to compute a max. matching in *general graphs*

## Better algorithms:

- Best known running time (bipartite and general gr.):  $O(m\sqrt{n})$

## Weighted matching:

- Edges have weight, find a matching of **maximum total weight**
- *Bipartite graphs*: **flow reduction** works ~~(in the same way)~~
- *General graphs*: can also be solved in polynomial time  
(Edmond's algorithms is used as blackbox)

# Happy Holidays!

