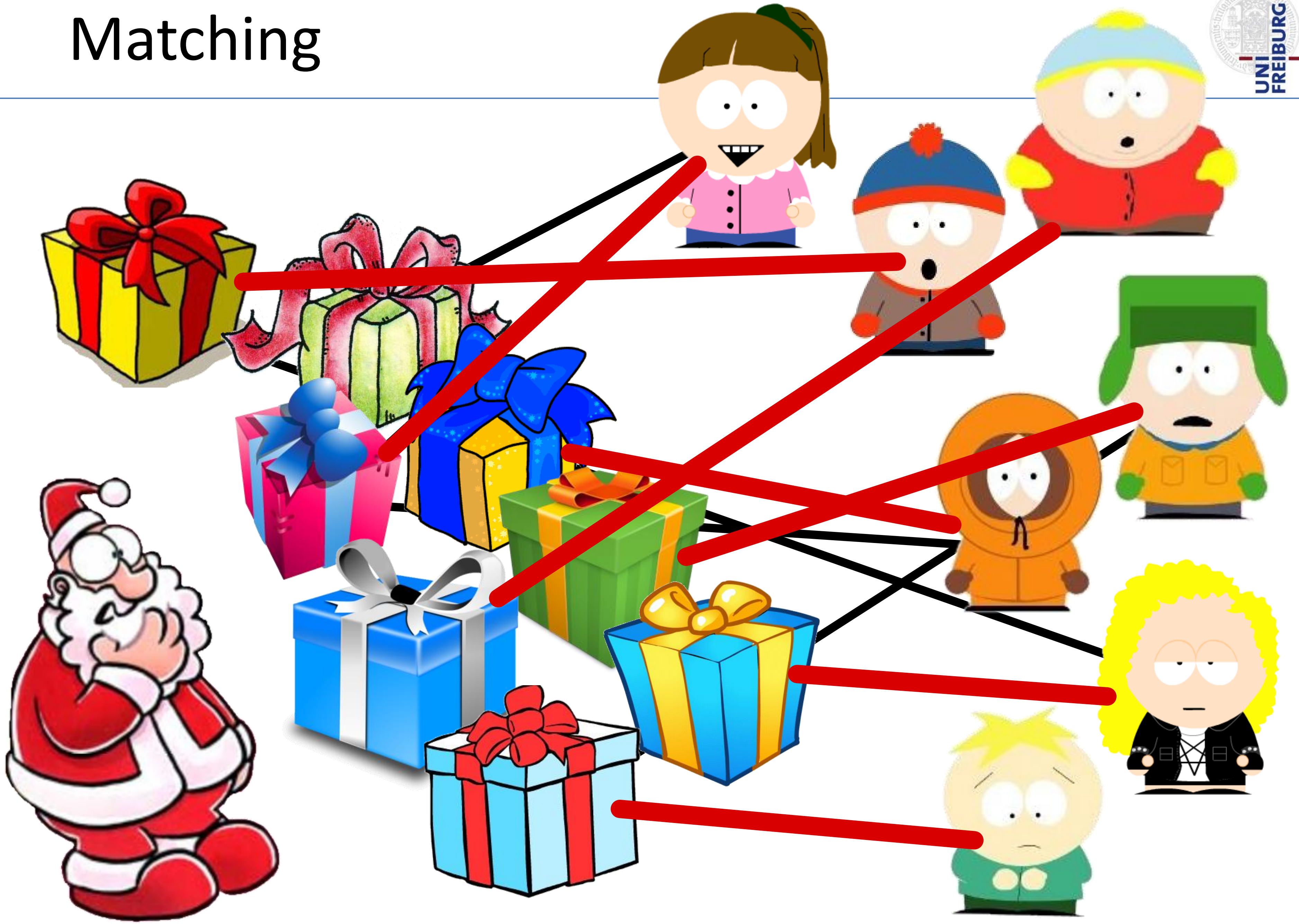
Chapter 6
Graph Algorithms

Algorithm Theory
WS 2017/18

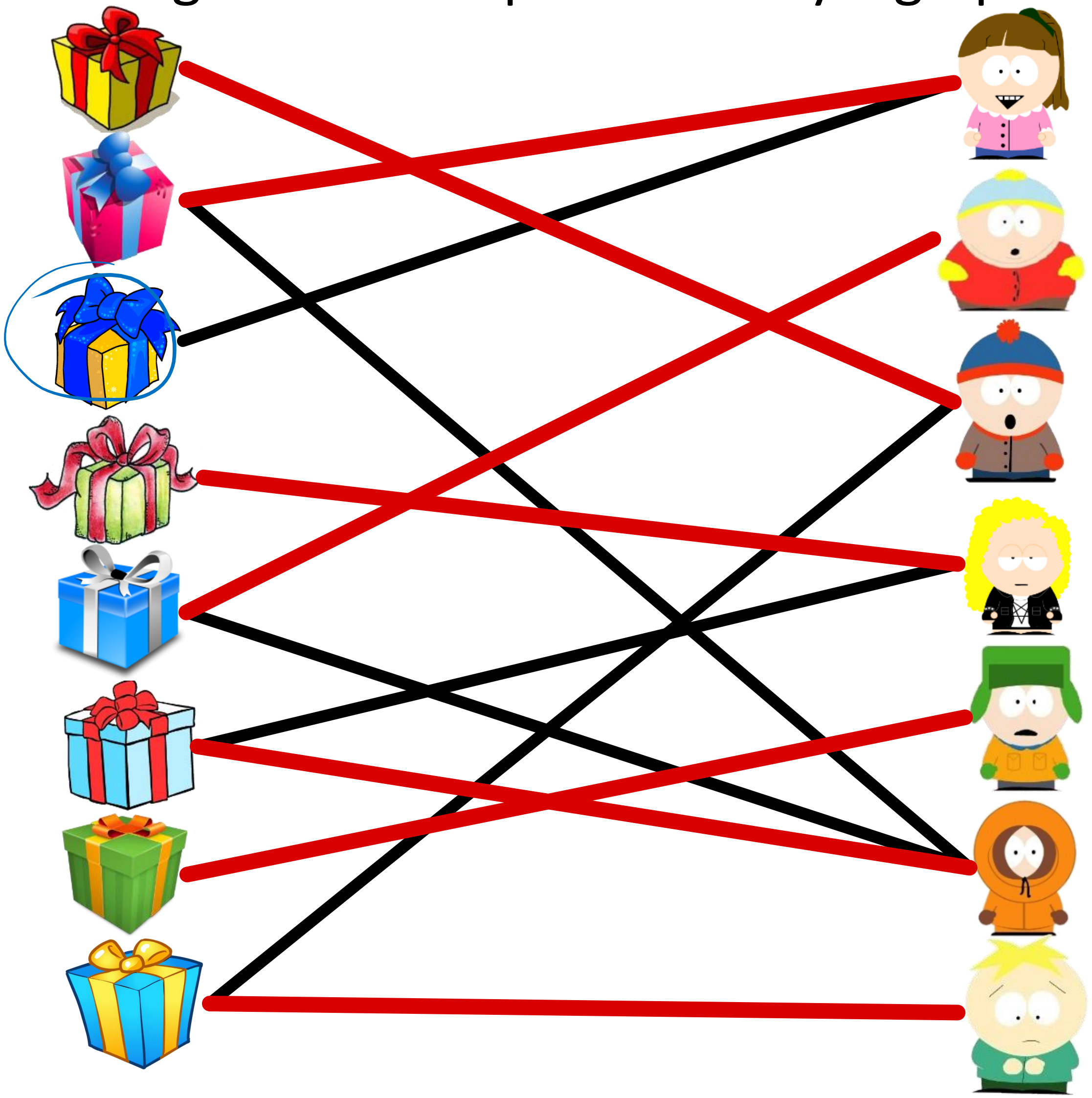
Fabian Kuhn

Matching



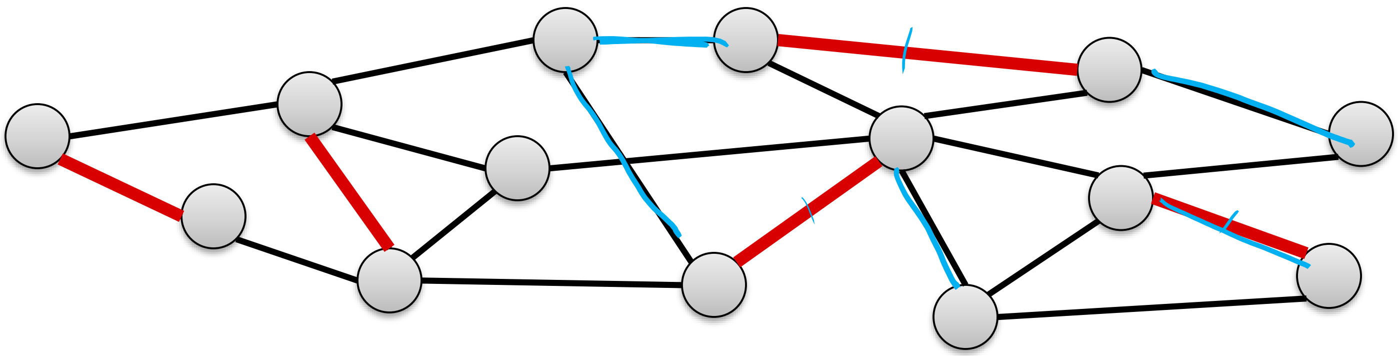
Gifts-Children Graph

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



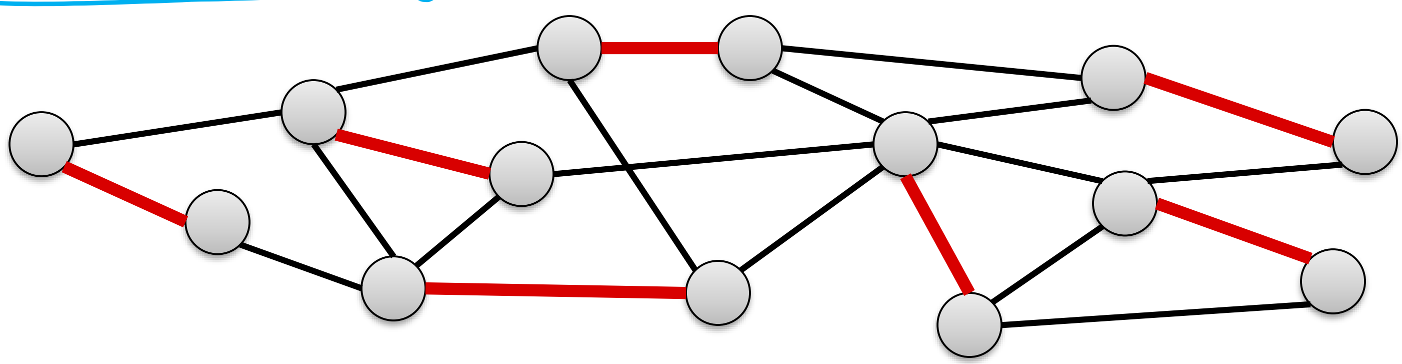
Matching

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



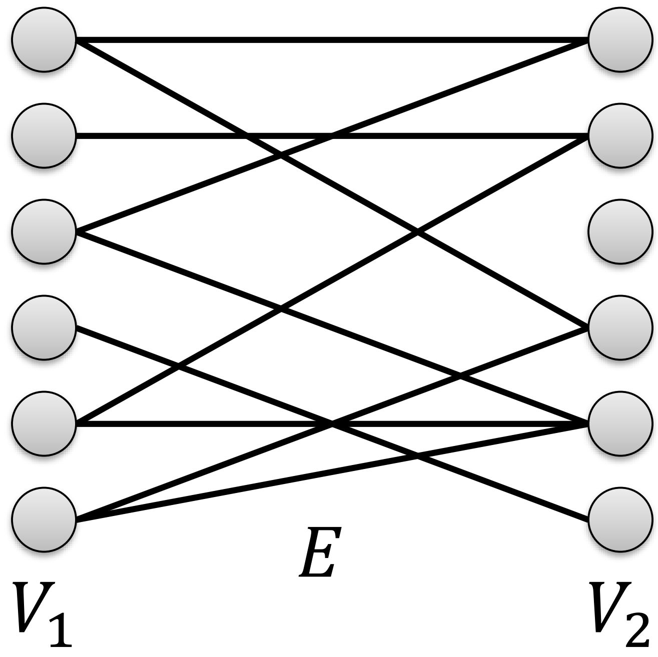
Perfect Matching: Matching of size $n/2$ (every node is matched)

Bipartite Graph

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

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

- Thus, edges are only between the two parts



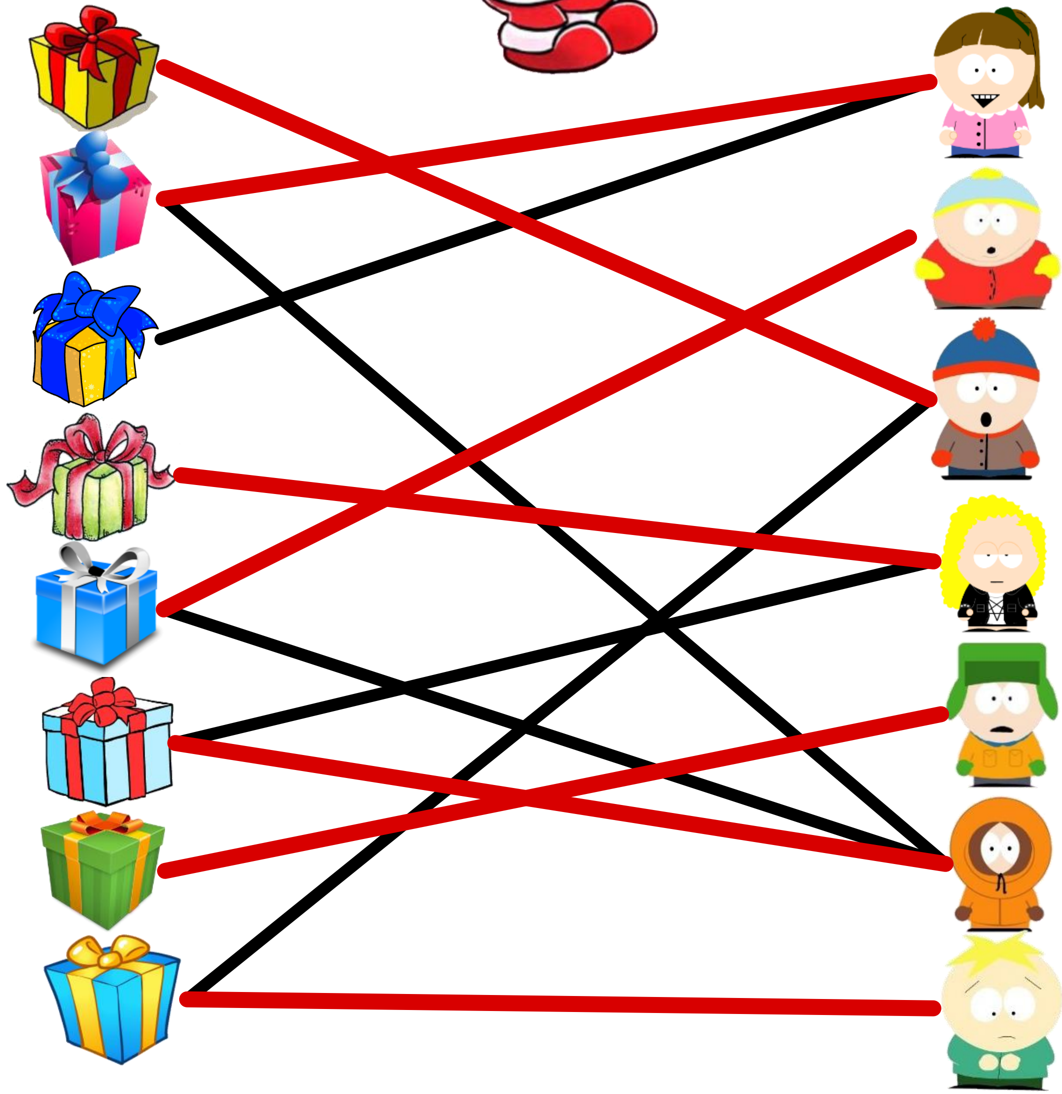
Santa's Problem

Maximum Matching in Bipartite Graphs:

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

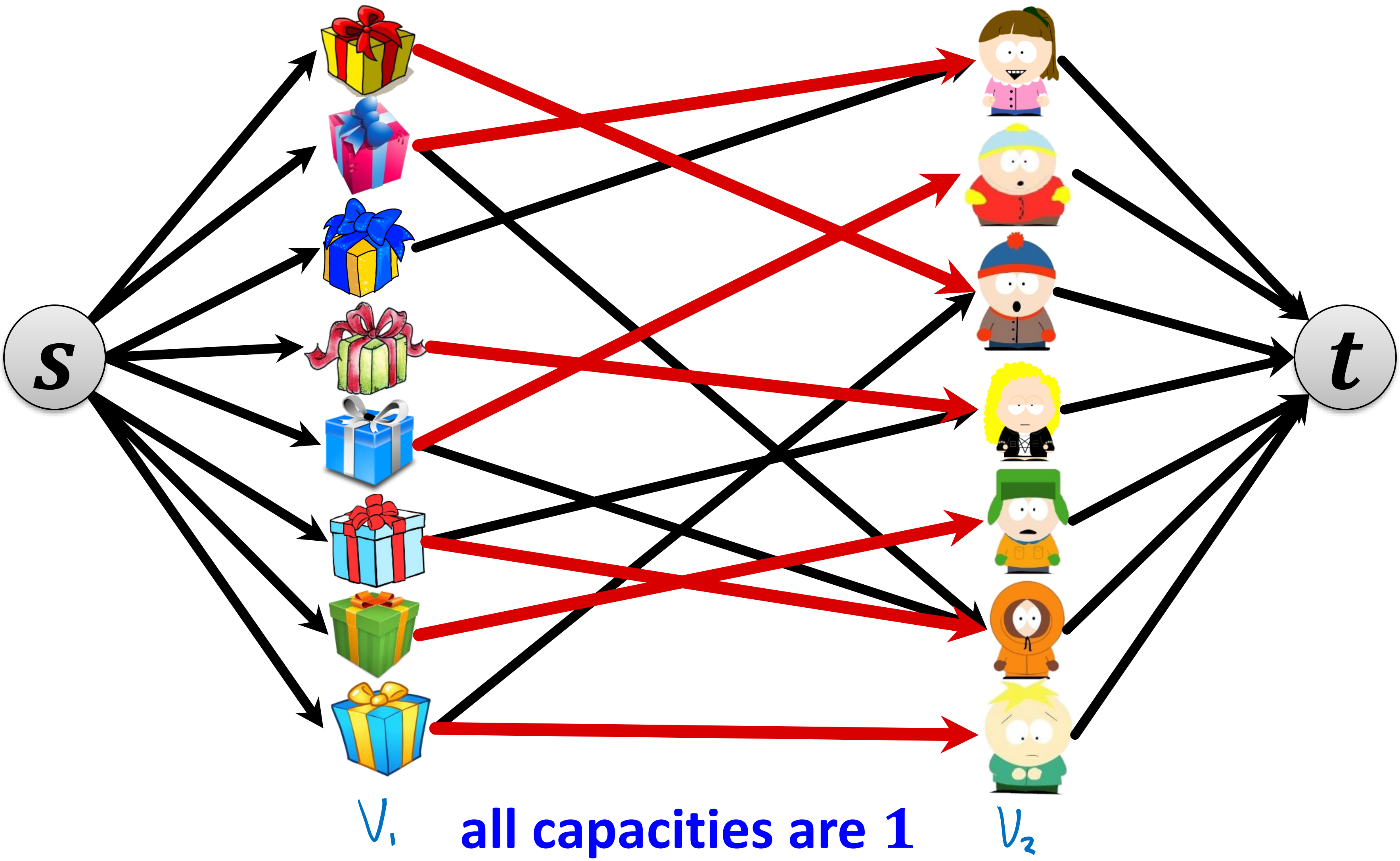
Clearly, every matching is at most as big

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



Reducing to Maximum Flow

- Like edge-disjoint paths…



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 ≤ 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:

$$f\big((s,u)\big) = f\big((u,v)\big) = f\big((v,t)\big) = 1$$

   - 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 $O(m \cdot n)$.

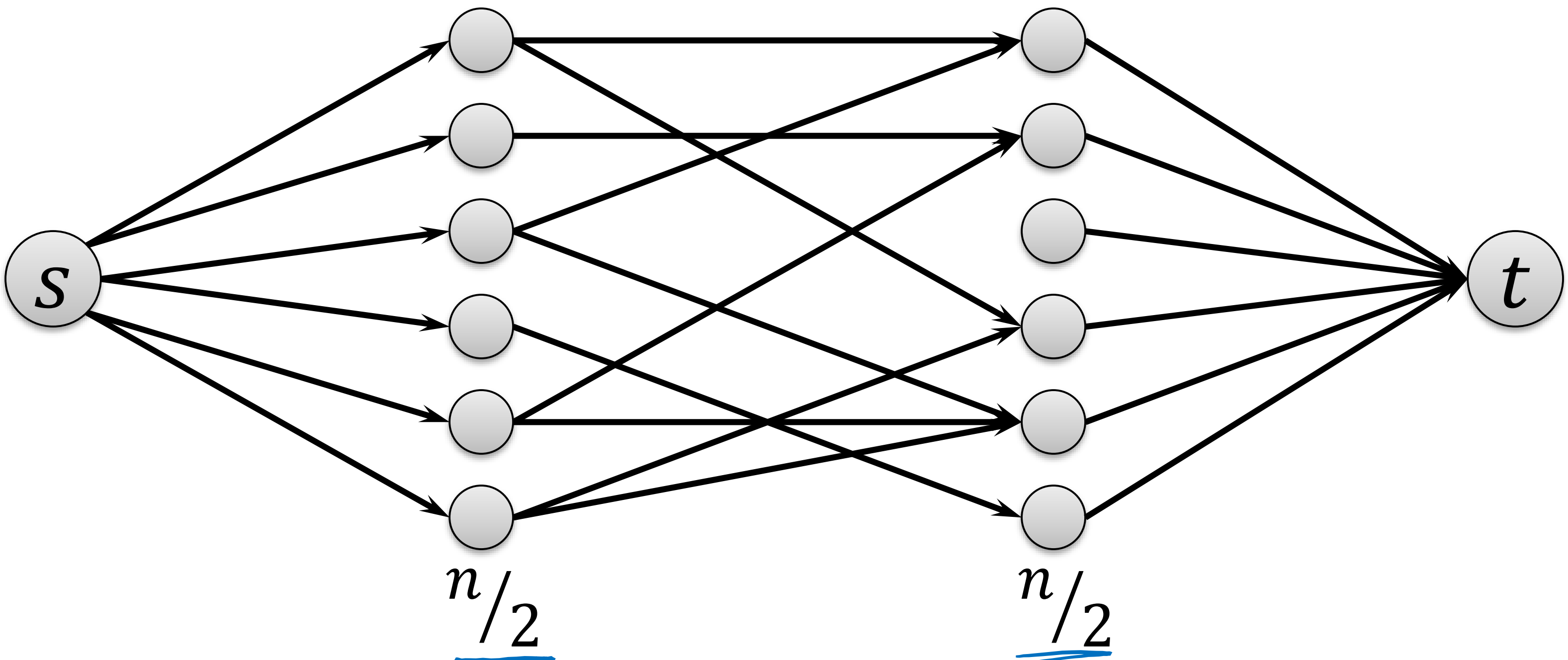
Ford Fulkerson

$$O(m \cdot C) = O(m \cdot n)$$

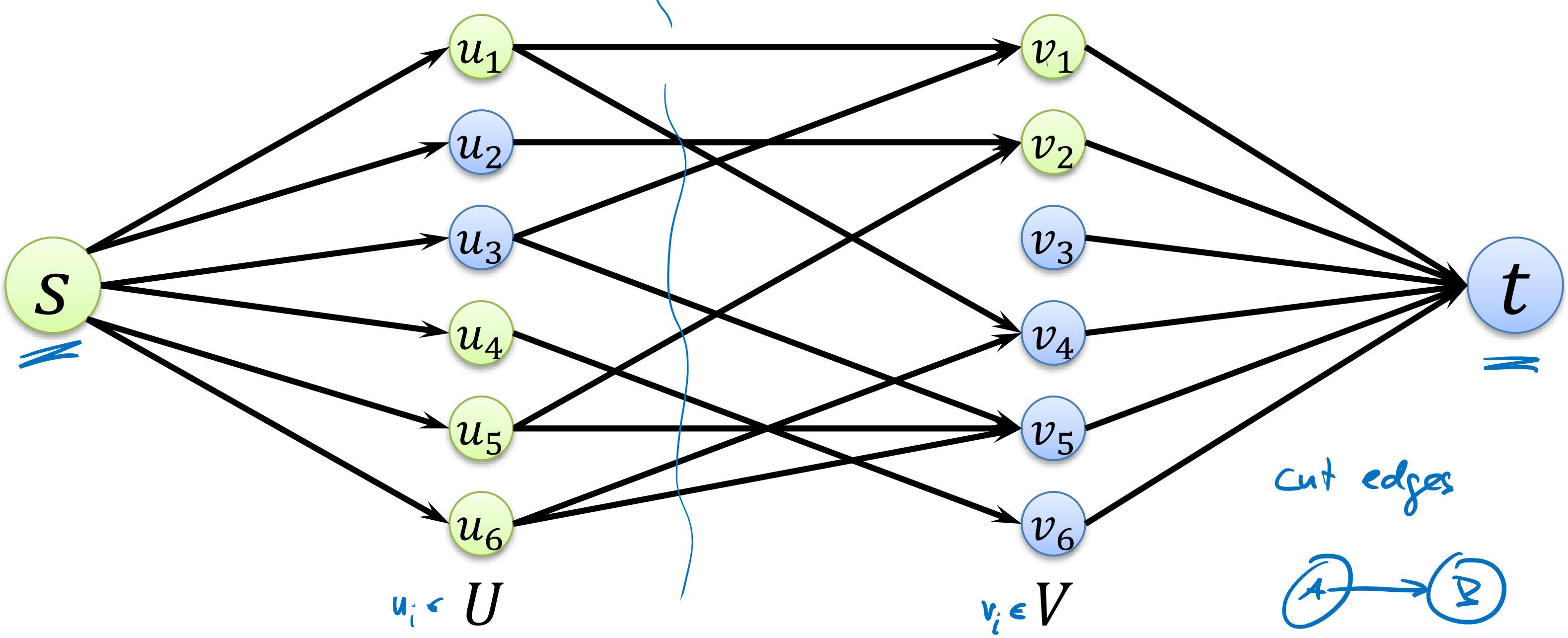
Perfect Matching?

- 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

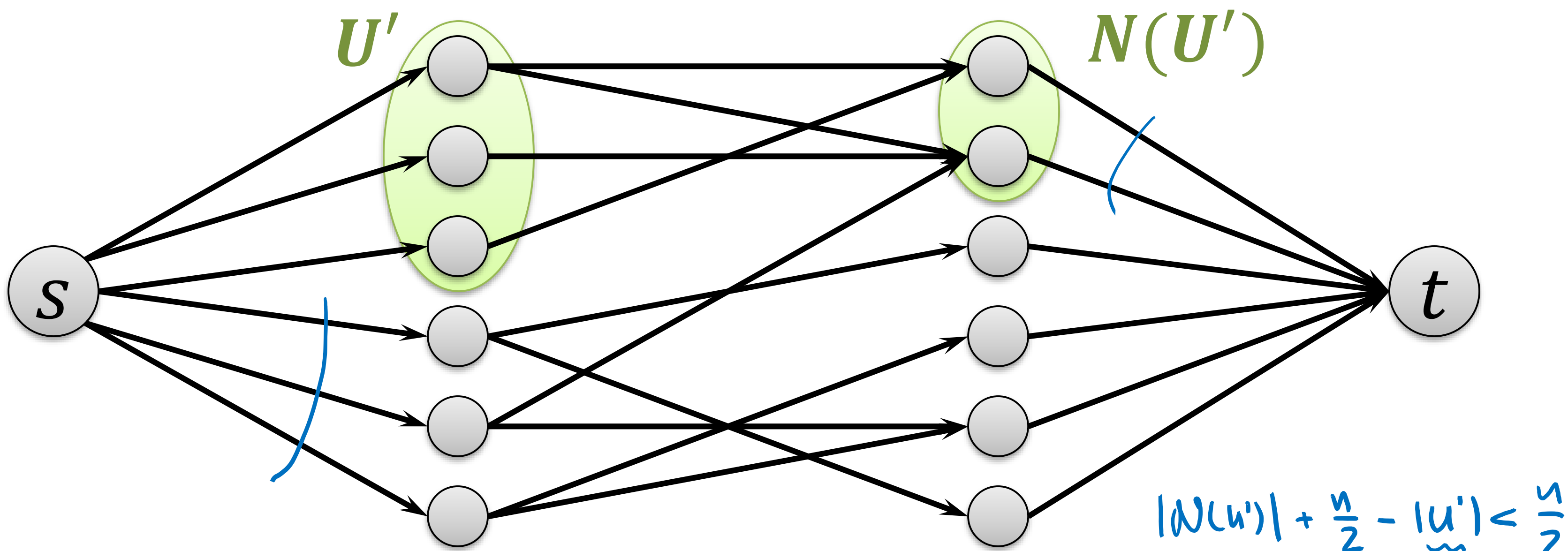
Theorem: A bipartite graph $G = (U \cup V, E)$ for which $|U| = |V|$ has a perfect matching if and only if

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

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 $|N(U')| < |\mathrm{U}'|$:



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

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

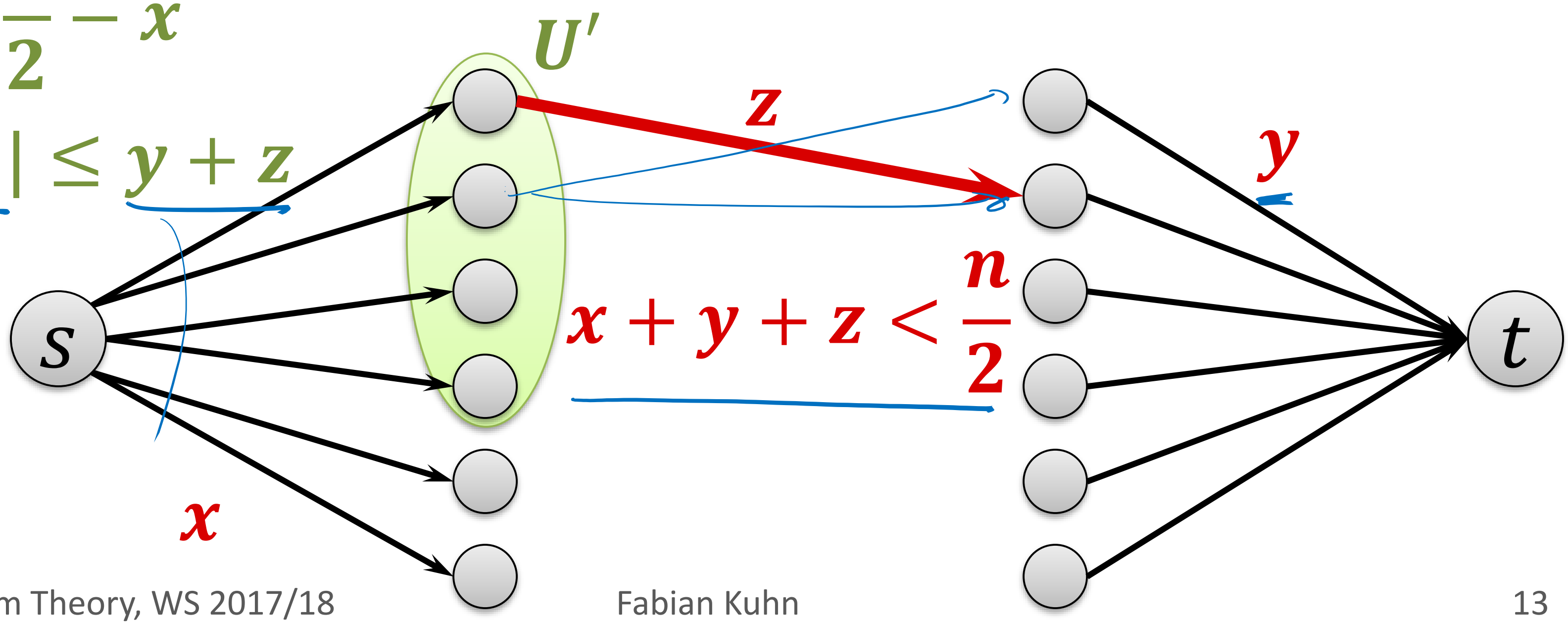
where $N(U') \subseteq V$ is the set of neighbors of nodes in U'.

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

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

$$|U'| = \frac{n}{2} - x$$

$$|N(U')| \leq y + z$$



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 $< n$

$$|U'| = \frac{n}{2} - x$$

$$|N(U')| \leq y + z$$

$$x + y + z < \frac{n}{2}$$

$$\underbrace{\frac{n}{2} - x}_{=|U'|} > y + z = |N(U')|$$

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…

- **Theorem:** The size of any maximal matching is at least half the size of a maximum matching.

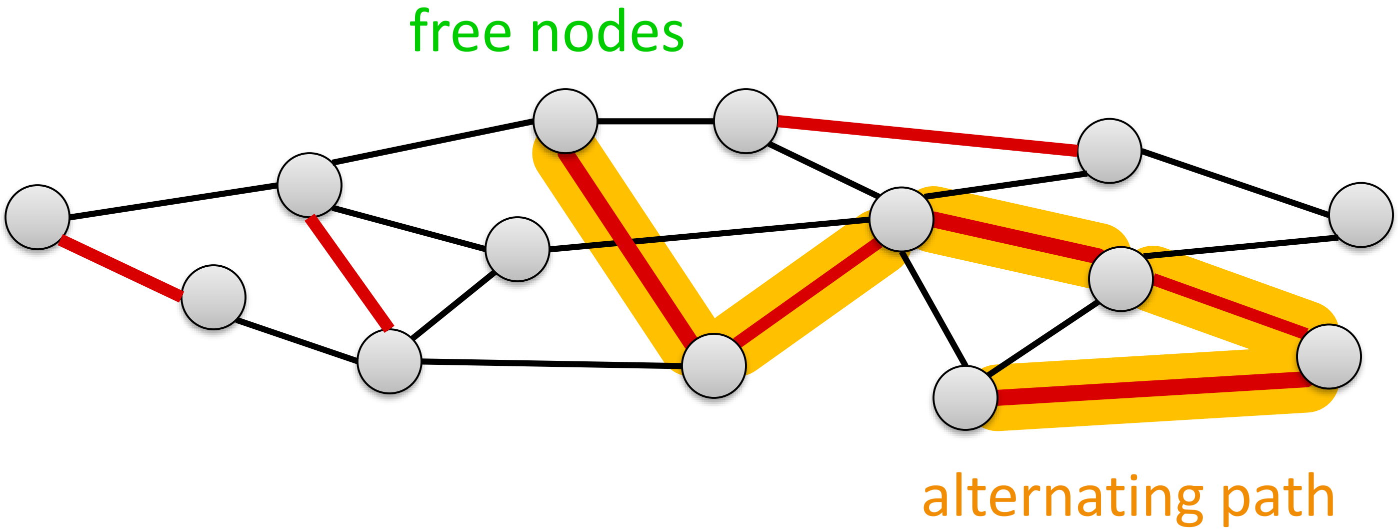
  - See next exercise sheet!
    (even for a natural generalization to the weighted version of the problem)

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

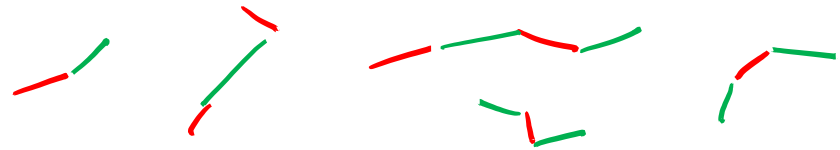
Augmenting Path: A (odd-length) path that starts and ends at a free node and visits edges in $E \setminus M$ and edges in M alternatingly.



- 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.

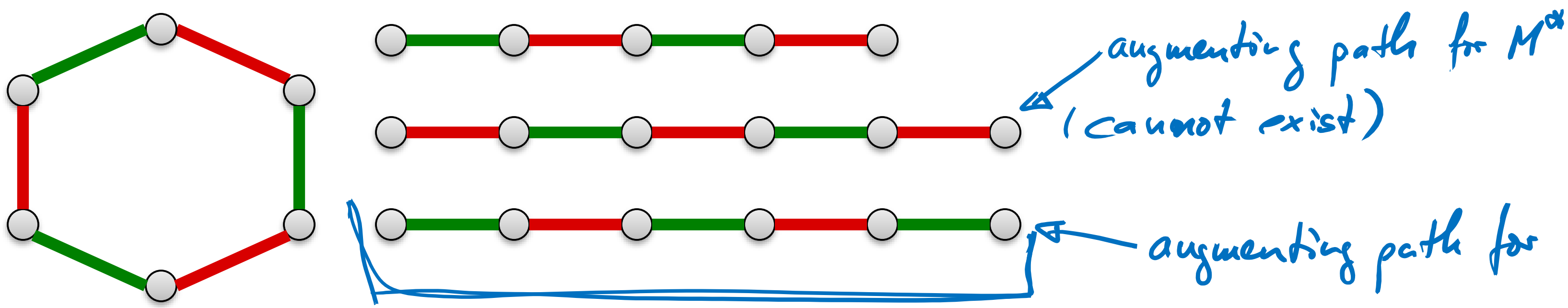


Proof:

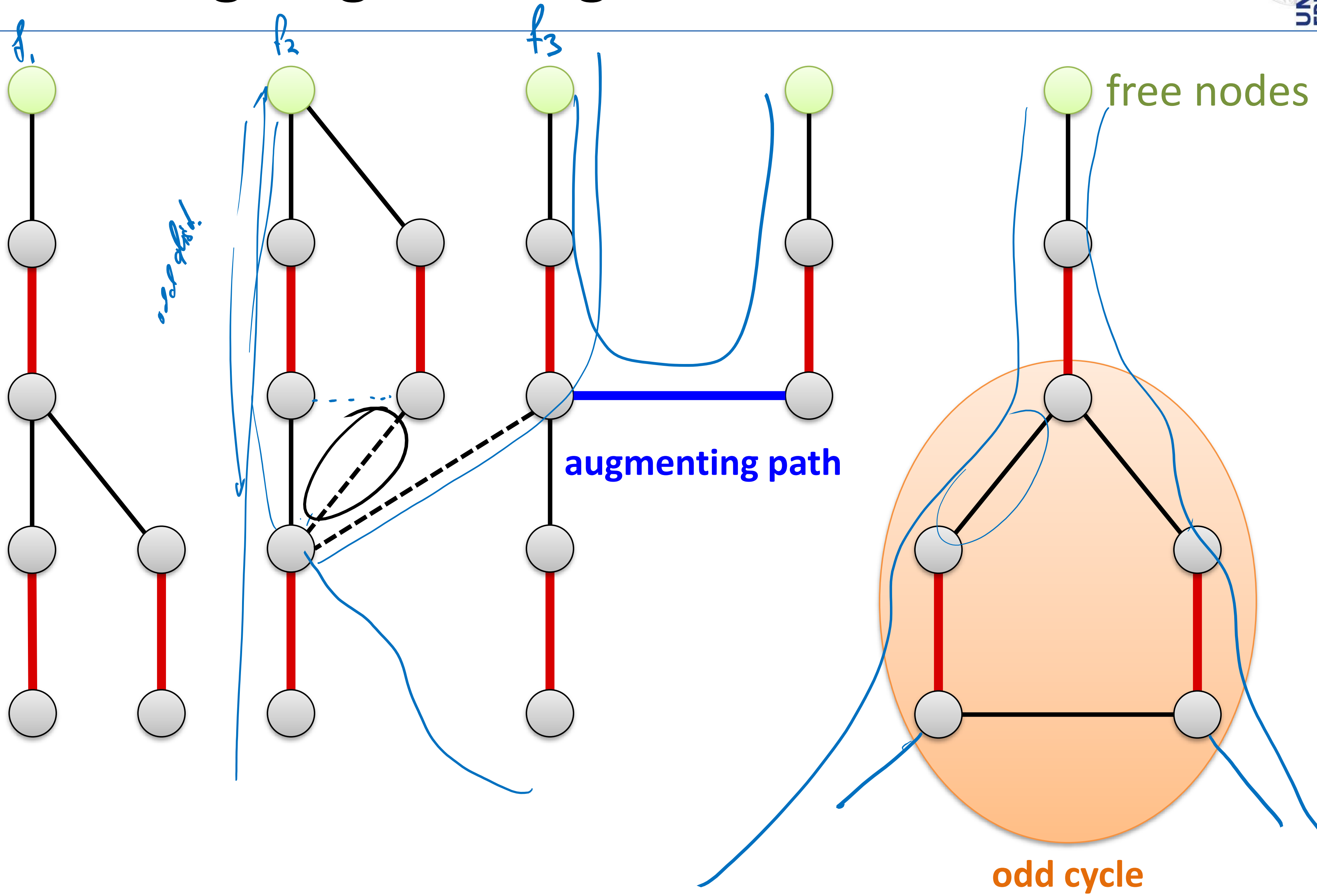
- Consider non-max. matching M and max. matching M^* and define

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

- Note that $F \cap F^* = \emptyset$ and $|F| < |F^*|$
- 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

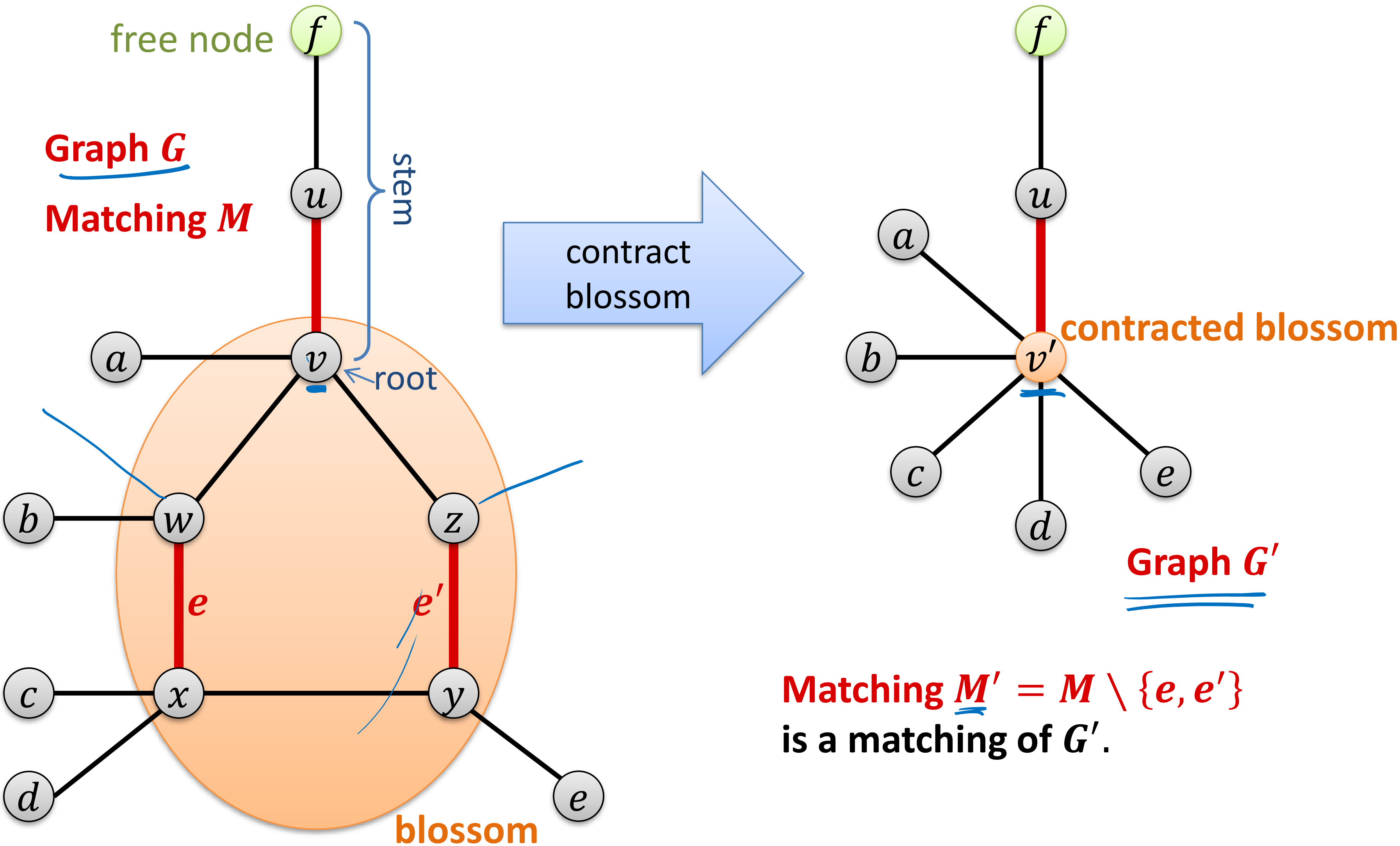


Finding Augmenting Paths



Blossoms

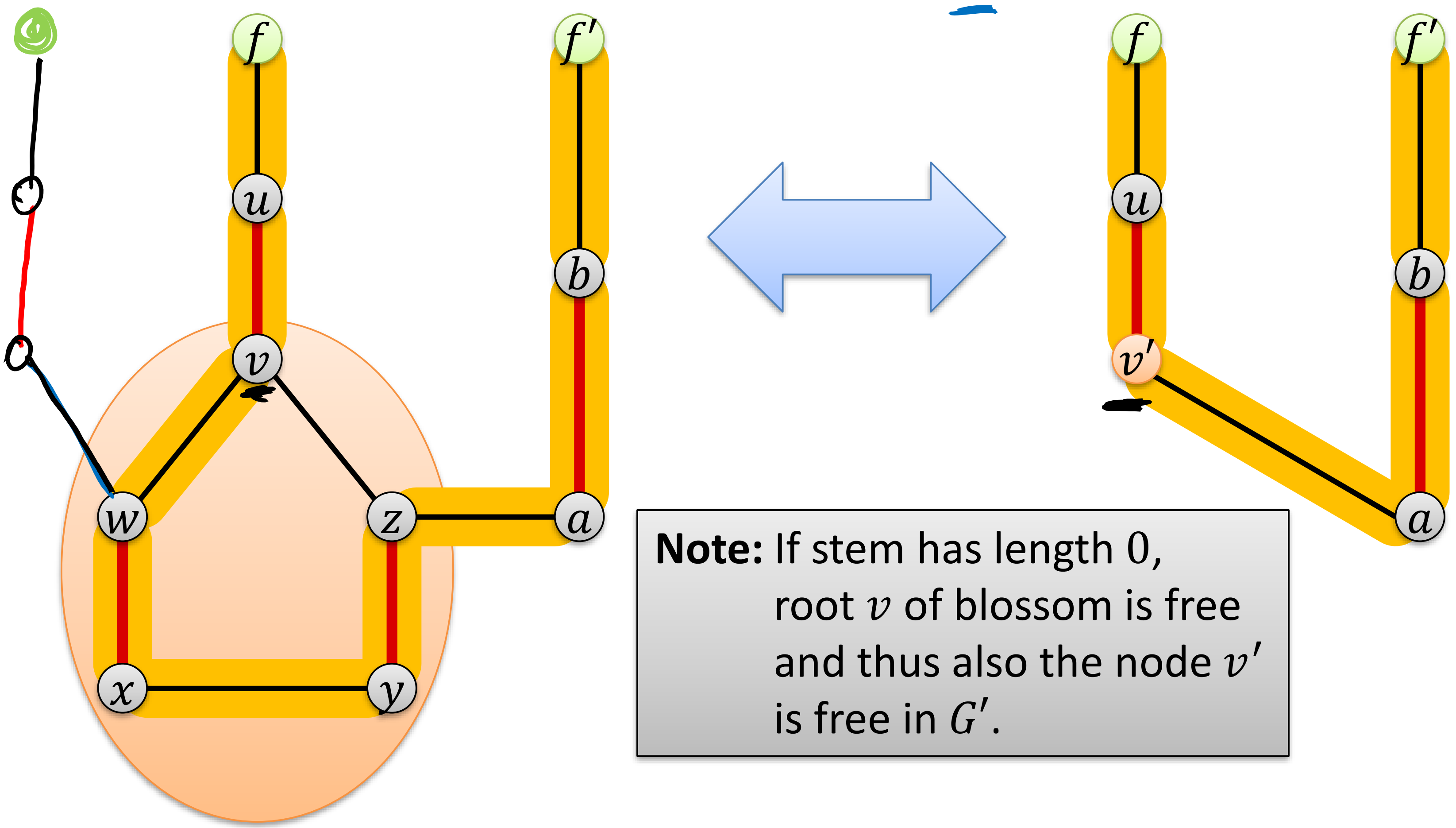
- If we find an odd cycle…



Matching $M' = M \setminus \{e, e'\}$ is a matching of G'.

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 → ignore and move on
       at even distance from root → **blossom found** contract blossom, continue on G'
    3. If v is explored and in another tree
       at odd distance from root → ignore and move on
       at even distance from root → **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 $O(mn^2)$.

graph expl. to find augm. path or blossom : DFS traversal

time: $O(m)$

can contract only $O(n)$ blossoms until find augm. path

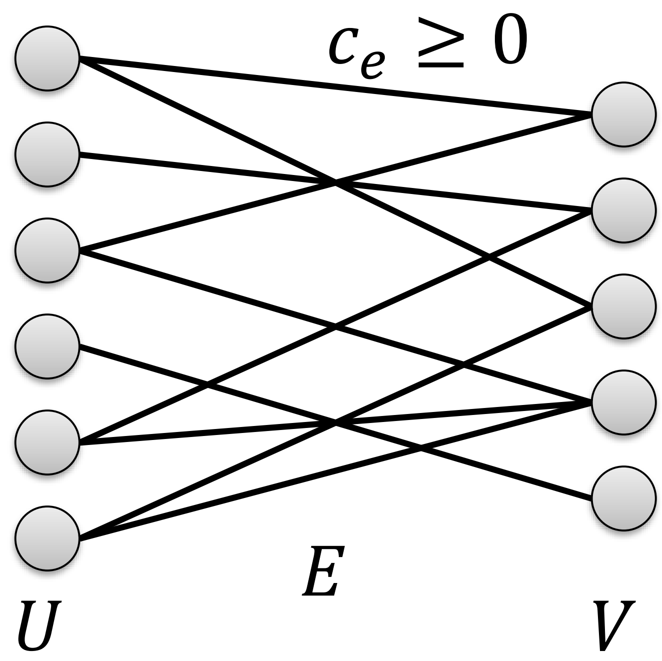
at most $n/2$ augm. paths

Maximum Weight Bipartite Matching

- Let's again go back to bipartite graphs…

Given: Bipartite graph $G = (U \dot{\cup} V, E)$ with edge weights $c_e \geq 0$

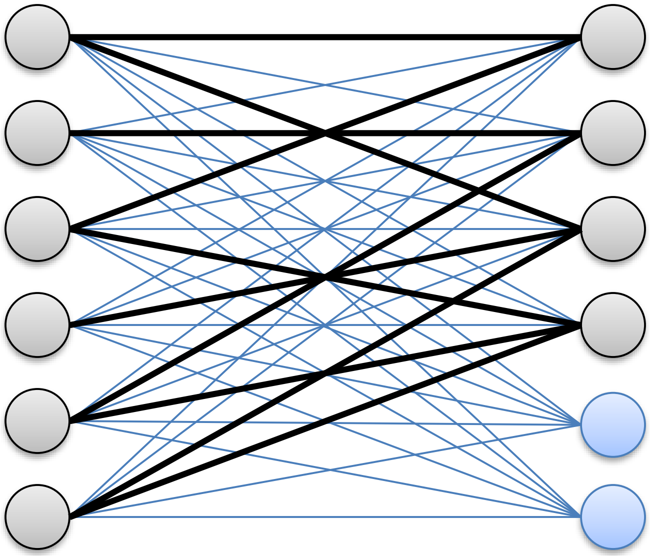
Goal: Find a matching M of maximum total weight



Minimum Weight Perfect Matching

Claim: Max weight bipartite matching is equivalent to finding a minimum weight perfect matching in a complete bipartite graph.

1. Turn into maximum weight perfect matching
   - add dummy nodes to get two equal-sized sides
   - add edges of weight 0 to make graph complete bipartite
2. Replace weights: $c'_e := \max_f\{c_f\} - c_e$



As an Integer Linear Program

- We can formulate the problem as an integer linear program

Var. x_{uv} for every edge $(u, v) \in U \times V$ to encode matching M:

$$x_{uv} = \begin{cases} 1, & \text{if } \{u, v\} \in M \\ 0, & \text{if } \{u, v\} \notin M \end{cases}$$

Minimum Weight Perfect Matching

$$\min \sum_{(u,v) \in U \times V} c_{u,v} \cdot x_{u,v}$$

$$\forall u \in U: \quad \sum_{v \in V} x_{u,v} = 1$$

$$\forall v \in V: \quad \sum_{u \in U} x_{u,v} = 1$$

$$\forall u, v: \quad x_{u,v} \in \{0, 1\}$$

Linear Programming (LP) Relaxation

Linear Program (LP)

- Continuous optimization problem on multiple variables with a linear objective function and a set of linear side constraints

LP Relaxation of Minimum Weight Perfect Matching

- Weight c_{uv} & variable x_{uv} for ever edge $(u, v) \in U \times V$

$$\min \sum_{(u,v) \in U \times V} c_{uv} \cdot x_{uv}$$

$$\text{s.t.}$$

$$\forall u \in U: \sum_{v \in V} x_{uv} = 1,$$

$$\forall v \in V: \sum_{u \in U} x_{uv} = 1$$

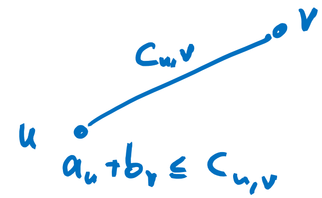
$$\forall u \in U, \forall v \in V: \; x_{uv} \geq 0$$

Dual Problem

- Every linear program has a dual linear program
  - The dual of a minimization problem is a maximization problem
  - Strong duality: primal LP and dual LP have the same objective value

In the case of the minimum weight perfect matching problem

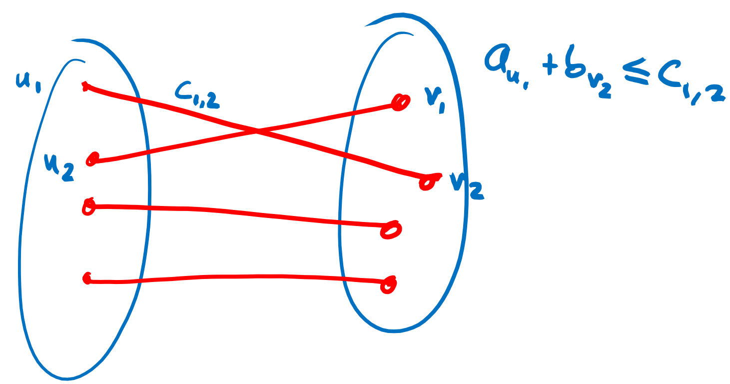
- Assign a variable $a_u \geq 0$ to each node $u \in U$
  and a variable $b_v \geq 0$ to each node $v \in V$



- **Condition:** for every edge $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{U} \times \boldsymbol{V}$: $\boldsymbol{a_u + b_v \leq c_{uv}}$

- Given perfect matching M:

$$\sum_{(u,v)\in M} c_{uv} \geq \sum_{u\in U} a_u + \sum_{v \in V} b_v$$



Dual Linear Program

- Variables $a_u \geq 0$ for $u \in U$ and $b_v \geq 0$ for $v \in V$

$$\max \sum_{u \in U} a_u + \sum_{v \in V} b_v$$

$$s.t.$$

$$\forall u \in U, \forall v \in V: \; a_u + b_v \leq c_{uv}$$

- For every perfect matching M:

$$\sum_{(u,v) \in M} c_{uv} \geq \sum_{u \in U} a_u + \sum_{v \in V} b_v$$

= would imply that M is optimal!

Complementary Slackness

- A perfect matching M is optimal if

$$\sum_{(u,v)\in M} c_{uv} = \sum_{u\in U} a_u + \sum_{v\in V} b_v$$

- In that case, for every $(u, v) \in M$

$$\boldsymbol{w_{uv}} \coloneqq c_{uv} - a_u - b_v = 0$$

  - In this case, M is also an optimal solution to the LP relaxation of the problem
  - Every optimal LP solution can be characterized by such a property, which is then generally referred to as complementary slackness

- **Goal:** Find a dual solution a_u, b_v and a perfect matching such that the complementary slackness condition is satisfied!
  - i.e., for every matching edge (u, v), we want $w_{uv} = 0$
  - We then know that the matching is optimal!

Algorithm Overview

- Start with any feasible dual solution a_u, b_v
  - i.e., solution satisfies that for all (u, v): $c_{uv} \geq a_u + b_v$

for example: $a_u = b_v = 0 \quad \forall u, v$

- Let E_0 be the edges for which $w_{uv} = 0$
  - Recall that $w_{uv} = c_{uv} - a_u - b_v$

- Compute maximum cardinality matching M of E_0

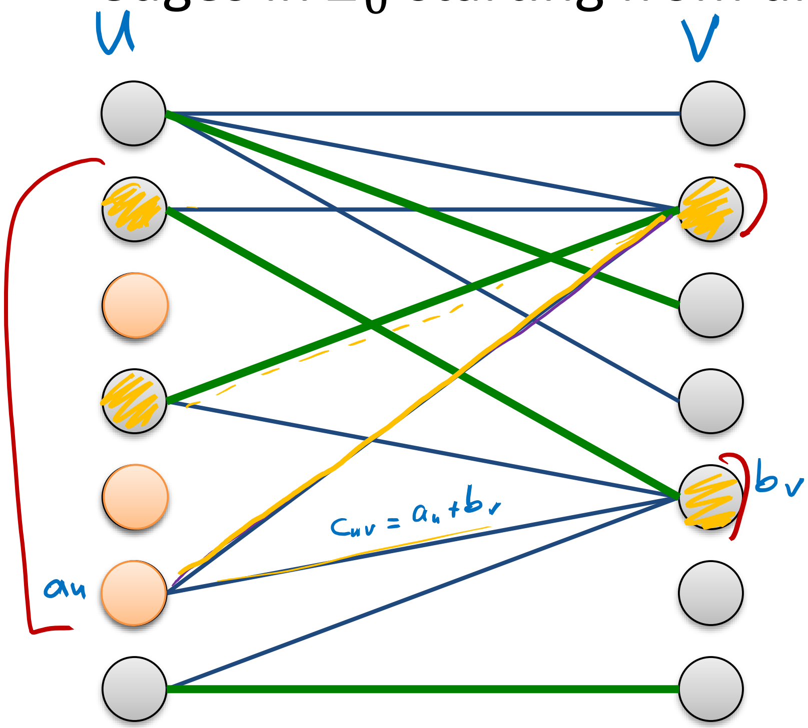
- All edges (u, v) of M satisfy $w_{uv} = 0$
  - Complementary slackness if satisfied
  - If M is a perfect matching, we are done

- If M is not a perfect matching, dual solution can be improved

Marked Nodes

Define set of marked nodes L:

- Set of nodes which can be reached on alternating paths on edges in E_0 starting from unmatched nodes in U



edges E_0 with $w_{uv} = 0$

optimal matching M

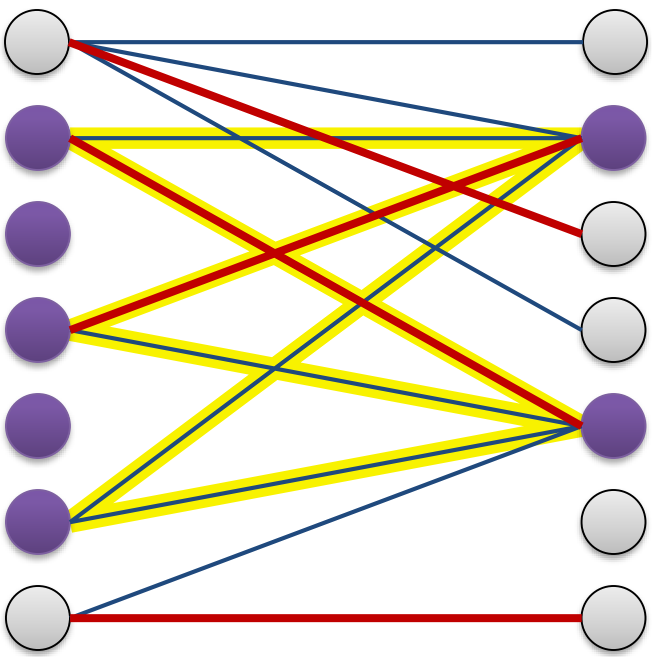
L_0: unmatched nodes in U

L: all nodes that can be reached on alternating paths starting from L_0

Marked Nodes

Define set of marked nodes L:

- Set of nodes which can be reached on alternating paths on edges in E_0 starting from unmatched nodes in U



edges E_0 with $w_{uv} = 0$

optimal matching M

L_0: unmatched nodes in U

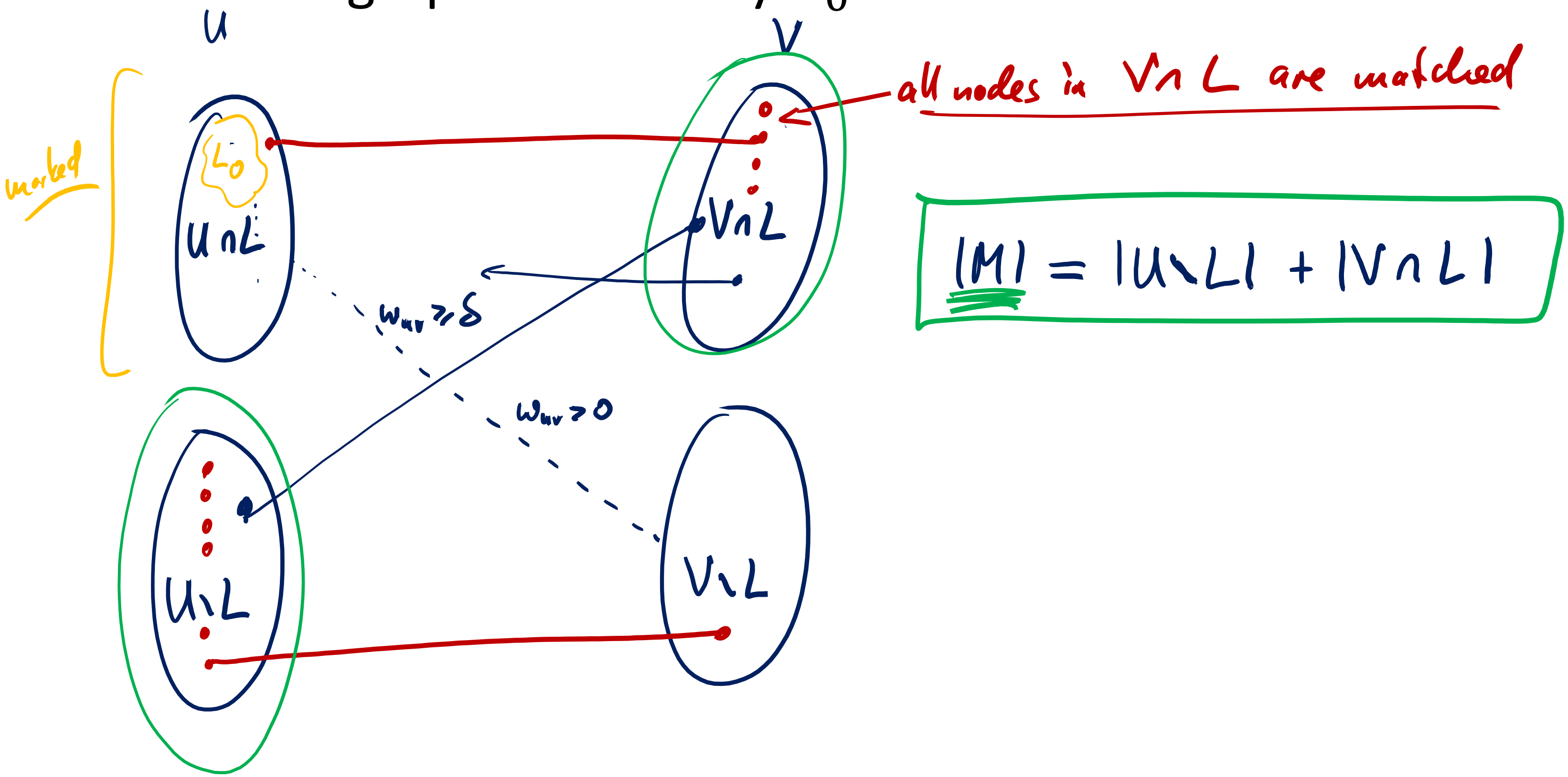
L: all nodes that can be reached on alternating paths starting from L_0

Marked Nodes – Vertex Cover

Lemma:

a) There are no E_0-edges between $U \cap L$ and $V \setminus L$

b) The set $(U \setminus L) \cup (V \cap L)$ is a vertex cover of size $|M|$ of the graph induced by E_0



Improved Dual Solution

Recall: all edges (u, v) between $U \cap L$ and $V \setminus L$ have $w_{uv} > 0$

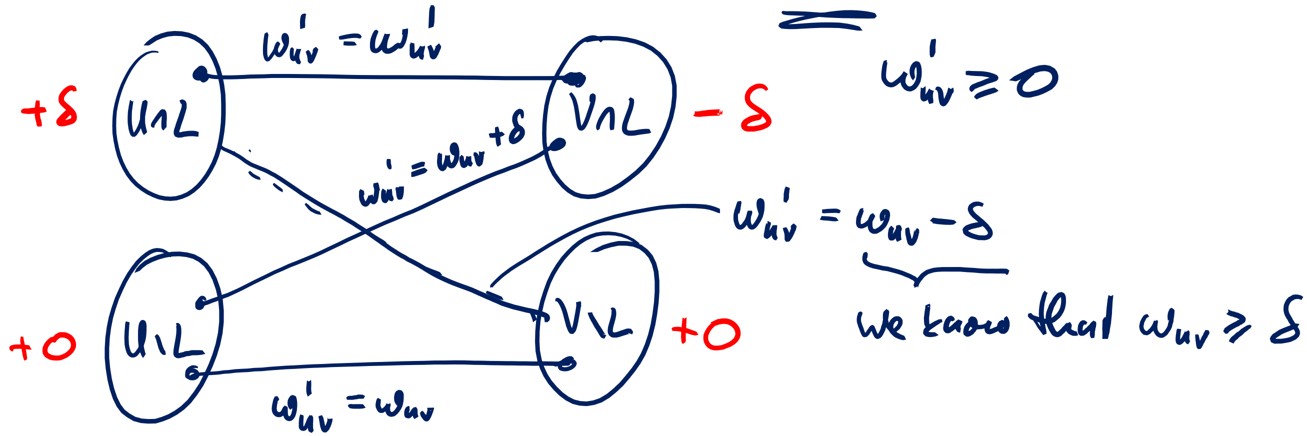
New dual solution:

$$\delta := \min_{u \in U \cap L,\, v \in V \setminus L} \{w_{uv}\} > 0 \qquad w'_{uv} = c_{uv} - a'_u - b'_v$$

$$a'_u := \begin{cases} a_u, & \text{if } u \in U \setminus L \\ a_u + \delta, & \text{if } u \in U \cap L \end{cases}$$

$$b'_v := \begin{cases} b_v, & \text{if } v \in V \setminus L \\ a_v - \delta, & \text{if } v \in V \cap L \end{cases}$$

Claim: New dual solution is feasible (all w_{uv} remain ≥ 0)



Improved Dual Solution

Lemma: Obj. value of the dual solution grows by $\delta\left(\frac{n}{2} - |M|\right)$.

Proof:

$$\delta := \min_{u \in U\cap L,\, v \in V\setminus L} \{w_{uv}\}, \qquad a'_u := \begin{cases} a_u, & \text{if } u \in U\setminus L \\ a_u + \delta, & \text{if } u \in U \cap L \end{cases}, \qquad b'_v := \begin{cases} b_v, & \text{if } v \in V\setminus L \\ a_v - \delta, & \text{if } v \in V \cap L \end{cases}$$

$$D = \sum_{u \in U} a_u + \sum_{v \in V} b_v \qquad D' = \sum_{u \in U} a'_u + \sum_{v \in V} b'_v$$

$$D' = D + \delta\big(|U\cap L| - |V \cap L|\big)$$

$$= D + \delta\Big(\underbrace{|U\cap L| + |U\setminus L|}_{=|U| = n/2} - \underbrace{\big(|V\cap L| + |U\setminus L|\big)}_{=|M|}\Big)$$

$$= D + \delta\left(\frac{n}{2} - |M|\right)$$