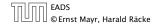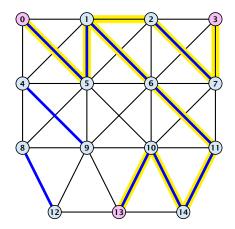# 18 Augmenting Paths for Matchings

Definitions.

▶ Given a matching M in a graph G, a vertex that is not incident to any edge of M is called a free vertex w. r. .t. M.

▶ For a matching M a path P in G is called an alternating path if edges in M alternate with edges not in M.

▶ An alternating path is called an augmenting path for matching M if it ends at distinct free vertices.

Theorem 1
A matching M is a maximum matching if and only if there is no augmenting path w. r. t. M.

Augmenting Paths in Action
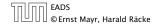
18 Augmenting Paths for Matchings

Proof.

⇒ If M is maximum there is no augmenting path P, because we could switch matching and non-matching edges along P. This gives matching $M' = M \oplus P$ with larger cardinality.

⇐ Suppose there is a matching M' with larger cardinality. Consider the graph H with edge-set $M' \oplus M$ (i.e., only edges that are in either M or M' but not in both).

Each vertex can be incident to at most two edges (one from M and one from M'). Hence, the connected components are alternating cycles or alternating path.

As $|M'| > |M|$ there is one connected component that is a path P for which both endpoints are incident to edges from M'. P is an alternating path.

18 Augmenting Paths for Matchings

Algorithmic idea:

As long as you find an augmenting path augment your matching using this path. When you arrive at a matching for which no augmenting path exists you have a maximum matching.
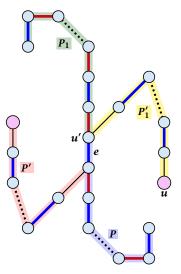
Theorem 2

Let G be a graph, M a matching in G, and let u be a free vertex w.r.t. M. Further let P denote an augmenting path w.r.t. M and let $M' = M \oplus P$ denote the matching resulting from augmenting M with P. If there was no augmenting path starting at u in M then there is no augmenting path starting at u in M'.
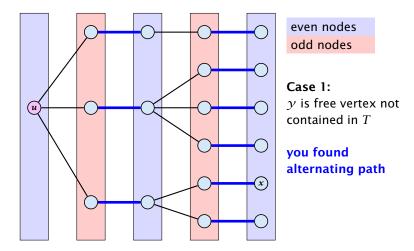
The above theorem allows for an easier implementation of an augmenting path algorithm. Once we checked for augmenting paths starting from u we don't have to check for such paths in future rounds.

18 Augmenting Paths for Matchings

Proof

▶ Assume there is an augmenting path P' w.r.t. M' starting at u.

▶ If P' and P are node-disjoint, P' is also augmenting path w.r.t. M (↯).

▶ Let u' be the first node on P' that is in P, and let e be the matching edge from M' incident to u'.

▶ u' splits P into two parts one of which does not contain e. Call this part P_1. Denote the sub-path of P' from u to u' with P_1'.

▶ $P_1 \circ P_1'$ is augmenting path in M (↯).

How to find an augmenting path?

Construct an alternating tree.

even nodes
odd nodes

Case 1:
y is free vertex not contained in T

you found alternating path

How to find an augmenting path?

Construct an alternating tree.

even nodes
odd nodes

Case 2:
y is matched vertex
not in T; then
mate$[y] \notin T$

grow the tree

How to find an augmenting path?

Construct an alternating tree.

even nodes
odd nodes

Case 3:
y is already contained in T as an odd vertex

ignore successor y

How to find an augmenting path?

Construct an alternating tree.

even nodes
odd nodes

Case 4:
y is already contained in T as an even vertex

can't ignore y

does not happen in bipartite graphs

Algorithm 50 BiMatch(G, *match*)
1: **for** $x \in V$ **do** *mate*$[x] \leftarrow 0$;
2: $r \leftarrow 0$; *free* $\leftarrow n$;
3: **while** *free* ≥ 1 **and** $r < n$ **do**
4:     $r \leftarrow r + 1$
5:     **if** *mate*$[r] = 0$ **then**
6:         **for** $i = 1$ **to** m **do** *parent*$[i'] \leftarrow 0$
7:         $Q \leftarrow \emptyset$; Q. append(r); *aug* $\leftarrow$ false;
8:         **while** *aug* = false **and** $Q \neq \emptyset$ **do**
9:             $x \leftarrow Q$. dequeue();
10:             **for** $y \in A_x$ **do**
11:                 **if** *mate*$[y] = 0$ **then**
12:                     augm(*mate*, *parent*, y);
13:                     *aug* $\leftarrow$ true;
14:                     *free* $\leftarrow$ *free* $- 1$;
15:                 **else**
16:                     **if** *parent*$[y] = 0$ **then**
17:                         *parent*$[y] \leftarrow x$;
18:                         Q. enqueue(*mate*$[y]$);

graph $G = (S \cup S', E)$

$S = \{1, \dots, n\}$

$S' = \{1', \dots, n'\}$

start with an
empty matching

free: number of
unmatched nodes in
S

r: root of current tree

as long as there are
unmatched nodes and
we did not yet try to
grow from all nodes we
continue