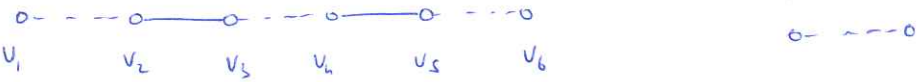
Lecture 14: Finding Matchings in General graphs.

Last time: Augmenting path:

Let M be a matching. A path $p = (v_1, \ldots, v_k)$ of odd length (i.e., even # vertices) is an augmenting path if:

(i) v_1, v_k are exposed vertices

(ii) alternate edges are in M

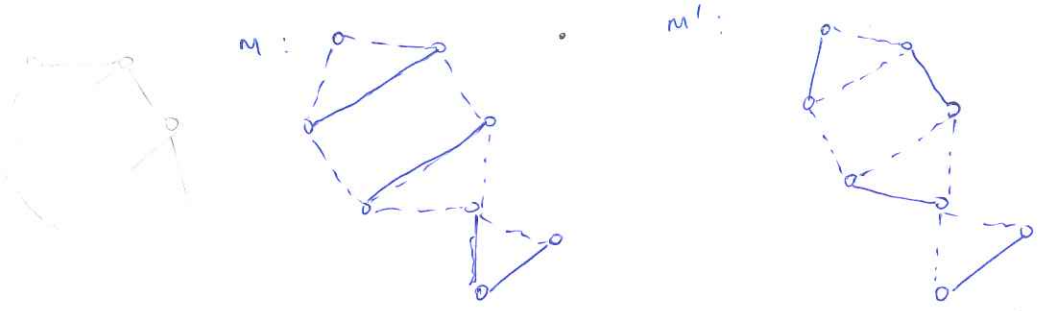


Berge's lemma: For any graph G, M is a maximum matching iff there is no a.p.

Last time we proved for b.p. graphs, giving an algorithmic proof. Now for general graphs.

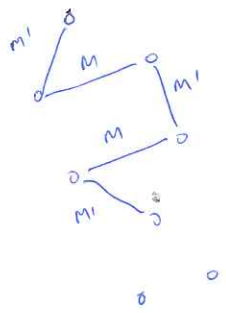
Proof: one direction is easy: let M be a matching, & $p = (v_1, \ldots, v_k)$ be an augmenting path. Then we can construct a larger matching by "flipping" edges along p, so that v_1, v_k are now matched. Hence M is not maximum.

Now suppose M is not maximum, & M' is a larger matching. We will show $\exists$ a.p. for M.



Consider graph $\bar{G} = (V, \bar{E})$

where $e \in \bar{E}$ iff e is in exactly one of M, M'. Then:

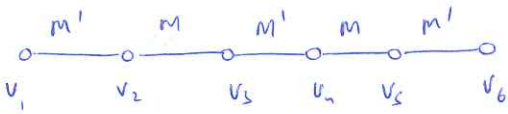


(I) every vertex has degree 0, 1 or 2

(II) $\bar{G}$ thus has isolated vertices, paths, cycles as components

(III) ~~In a cycle C, $C \cap M$~~ In a path or cycle, edges from $M, \bar{M}$ must alternate.

(IV) Since $|M'| > |M|$, $\exists$ a path p s.t. # edges in $p \cap M'$ $>$ $|p \cap M|$. Thus p looks like:

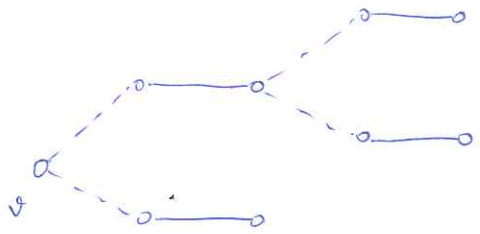


note that v_1, v_6 are exposed in M. Then p is an a.p.

Hence, if M not maximum, $\exists$ a.p. ∎

Last time we gave an algorithm that found augmenting paths in bipartite graphs by finding alternating trees:

Given a matching M (initially empty), let v be an exposed vertex.



we will run the same algorithm, slightly modified.

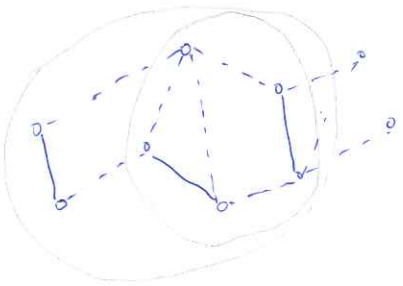
Earlier, no edges between vertices on the same level, since this would have meant an odd cycle. Now we are allowed odd cycles. How do we deal with this?

Let M be a matching. A blossom is an odd (simple) cycle C where:

(i) exactly 1 vertex is exposed

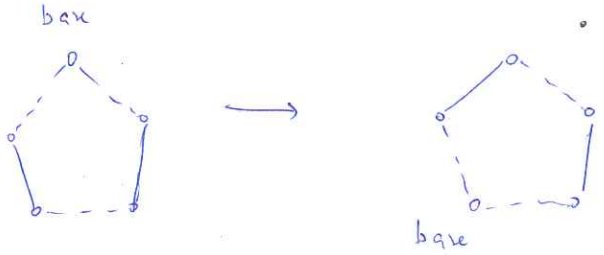
(ii) other vertices are matched by edges in the cycle

eg.



The exposed vertex is called the base of the blossom.

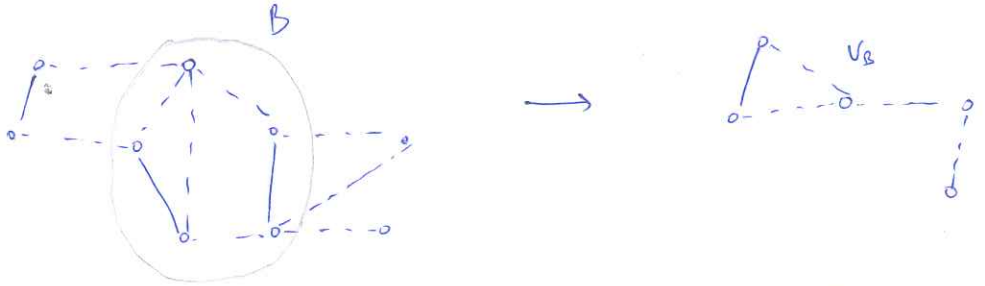
Note that given a matching M and blossom B, we can modify M so that any $v \in B$ is the base, since no vertex in B has a matching edge outside.



So why is a blossom good?

Let given graph G with matching M, let B be a blossom.

Let G' be obtained by contracting B to a single (exposed) vertex v_B, and removing multiple edges.



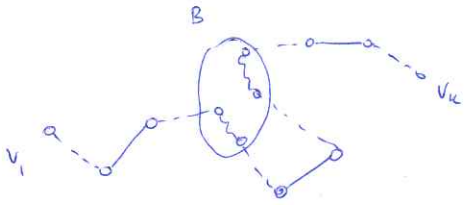
G $\qquad$ G'

Let M' be the matching in G'.

Then:

Claim: G (with M) has a a.p. iff G' (with M') has an a.p.

Proof: Let p be an a.p. in G that contains edges from B.



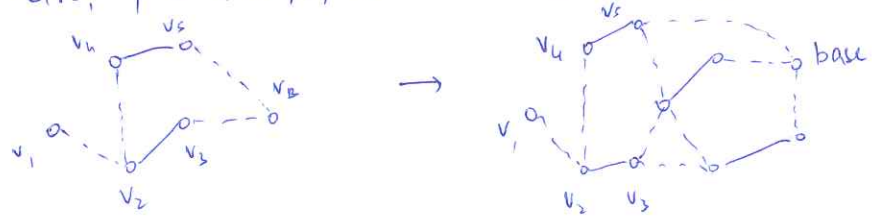
① Let $p = (v_1, \dots v_k)$ be the a.p. Then v_1, v_k are exposed vertices. Hence at most one of these is in B. Say $v_1 \notin B$.

Let v_r be first vertex in B. Since B is replaced by exposed vertex v_B, $p' = (v_1, v_2, \dots, v_B)$ is an a.p. in G'.

(vr earlier)

Now let p' be an a.p. in G'. If $v_B \notin p'$, then p' exists in G also.

Else, $p' = (v_1, \dots v_B)$ since v_B is exposed in G'.



Let v_r be vertex in p' just before v_0. Then in G, v_r has an edge to some vertex in B, say v_{r+1}. If v_{r+1} is base of B, then $v_1, \ldots, v_r, v_{r+1}$ is a.p. Else, follow alternating path in B to reach base, this gives an augmenting path. ∎

(i.e., follow ~~no~~ edge in M from v_{r+1}, the unmatched edge in B from v_{r+2}, then edge in M from v_{r+3}, ... until base of blossom is reached).

Our algorithm is now as follows.

Initially let $M = \phi$.

(*) Let v be an exposed vertex. Build alternating tree starting from v, with v even.

(A) If blossom found:
- contract blossom to a single exposed vertex. ~~[Go back to (*)]~~
- Let G' be modified graph. Run from (*) in G'.

(B) If augmenting path found:
- Find augmenting path in original graph G.
- Augment M to get larger matching.
- Run (*) in G.

Note: ① if last operation was (B).

$M \leftarrow \phi$, v is arbitrary vrtex)

Alt-Tree (G, M, v)

Alt-Tree $(\bar{G}, \bar{M}, v)$ { // v is exposed vertex in $\bar{G}$

Build alternating tree starting from v, with v.mark = even

if blossom B found:

- contract B to vertex v_B, modify $\bar{G}, \bar{M}$ to obtain G', M'
- run Alt-Tree (G', M', v_B)

if augmenting path p found:

~~[find augmenting path q in original graph G]~~

- expand blossom c in $\bar{G}$ to get original graph G with matching M.
- find augmenting q in G w/ matching M (exists because of claim)
- this gives large matching M' in G. If $\exists$ exposed vrtex v in G, run Alt-Tree (G, M', v), else stop. (A)

(starting from any exposed vrtex $w \in \bar{G}$)

if neither B nor p found, ~~(stop &)~~ print "maximum matching found," expand $\bar{M}$ to matching M in ~~M~~ G by expanding contracted vertices (B)

Claim: Suppose graph $\bar{G}, \bar{M}$, is passed to Alt-Tree. If $v_B \in \bar{G}$ is a contracted vertex, then v_B is exposed.

Proof: By induction, true for first call to Alt-Tree trivially. Suppose true for previous call. If blossom found, then matching is not changed, and newly contracted vrtex is exposed. If a.p. found, then $\bar{G}$ has no contracted vertices.

Claim: The algorithm returns a maximum matching in graph G.

Proof: Suppose algo stops with (A). Then there is no exposed vertex in G, and M must be a perfect matching.

Suppose algo stops with (B). Then there is no a.p. in $\bar{G}$. Hence by claim, there is no a.p. in G, and ~~algo~~ matching $\bar{M}$ in $\bar{G}$ is maximum. Hence expanding $\bar{M}$ gives maximum matching in G.

Claim: If $\exists$ an a.p. in $\bar{G}$ w/ matching $\bar{M}$ from vertex v, then Alt-Tree either ~~returns an~~ finds an a.p. or a blossom.

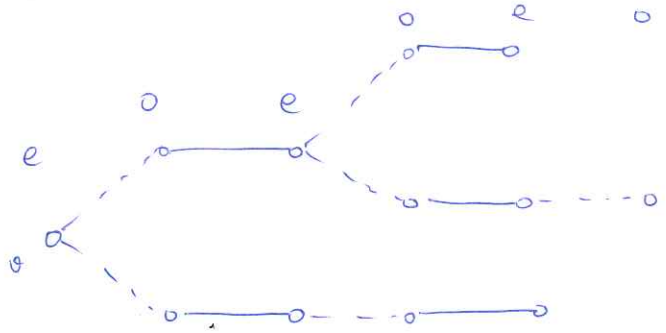
Proof: from algorithm.

- ✓ We can use the algorithm to obtain a characterization for graphs w/ perfect matching.
- ✓ **Hall's Theorem (for b.p graphs)**: graph $G = (A \cup B, E)$ has a perfect matching if $\forall S \subseteq A$, $|N(S)| \geq |S|$
- ✓ For general graphs, we have Tutte's theorem.

not crossed out. ✓

Suppose Alt-Tree $(\bar{G}, \bar{M}, v)$ can not find an a.p. or a blossom.

By claim, any contracted vertex are exposed.



for non-tree edges:

then: (I) no edges b/w vertices at ~~2 odd~~ levels that differ by ≥ 2

(II) for vertices on the same level:

- no ~~unmatched~~ edge b/w even vertices
- no matched edge b/w odd vertices

~~[(III) for vertices on levels one apart:]~~

~~[Thus, an even vertex only has edges]~~

(III) further, no vertex outside tree for even vertices

Thus, an even vertex only has edges to odd vertices.

Further: ~~(IV)~~ Claim: no leaf is odd.

Proof: suppose w is an odd leaf. Since v-w is not an augmenting path, w must be matched. Since w is odd, and is a leaf, it can only be matched to an even vertex. But all even vertices are already matched with tree edges.

Let T be the set of all tree vertices, T_E, T_O be even & odd vertices. Since root is even, and ~~apart from~~ unmatched, and every odd vertex is matched to an even vertex, $|T_O| < |T_E|$.

Consider the cut $(T_o, V \setminus T_o)$. Then:

- every ~~odd~~ even vertex in T is now an odd-sized component.

Hence, removal of T_o creates $> |T_o|$ odd components.

Thus, if ~~[M is not a perf]~~ G does not contain a perfect matching (v is exposed in M), then $\exists S \subseteq V$ s.t. removal of S creates $> |S|$ odd components.

This is a characterization for graphs with perfect matchings:

Tutte's Theorem: G has a p.m. iff $\forall S$, the graph $(G \setminus S)$ obtained by removing S and all edges incident to S, contains at most $|S|$ odd components.

Proof: We showed one direction. The other direction is easy.