max matching - Blossom's algorithm

Finding max matching

→ start with $m = \emptyset$

→ Find aug path P in (G, m)

→ if $P == [\,]$, return m, as max matching

→ else $m = m \oplus P$

Finding augmenting path in (G, m)

U = unmatched vertices

creating a forest with each tree being just a node

```
for each v ∈ U
    create a tree with root(v)
    label v as EVEN label
```

Forest F

mark each edge in $E \setminus m$ as unexplored, each edge in m as explored.

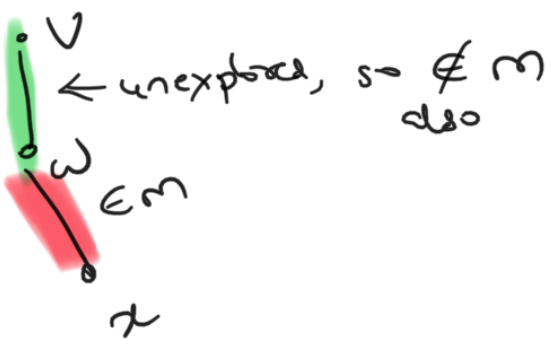
Queue = $\tilde{V}$ = vertices to be explored = $\bigcup$ ← will contain even labeled vertices

while $\tilde{V}$ is not empty:

$v = \tilde{V}$.pop()

for every unexplored edge $v - w$

$w \notin F$ (so w matched vertex)

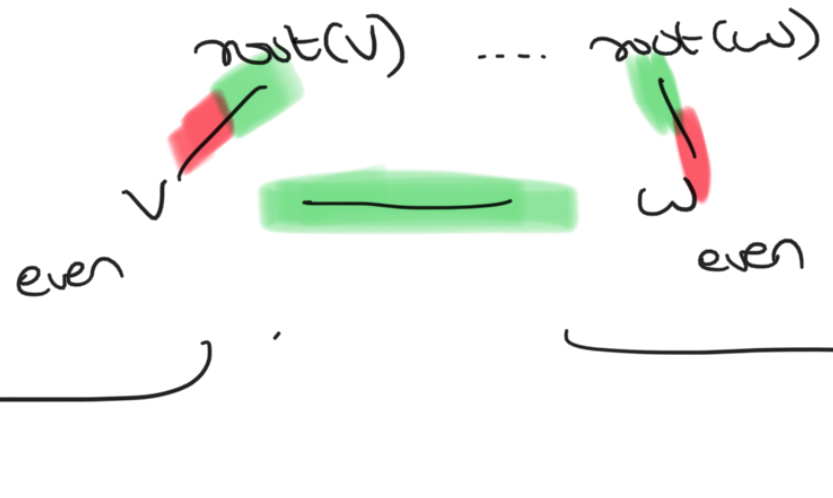


* label w odd
* label x even
* add $v-w$, $w-x$ to tree(v) (and hence to F)
* so $w, x \in F$ also now
* mark parent(x) = w
  parent(w) = v
* add x to $\tilde{V}$.

$w \in F$

mark $v-w$ as explored.

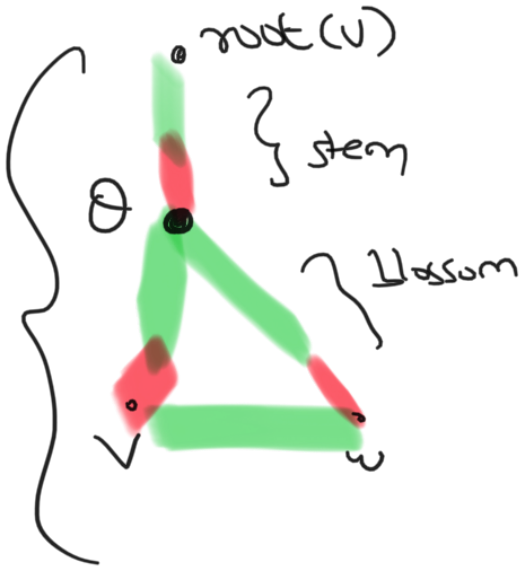
w has label even



root(v) ≠ root(w)

root(v) = root(w)

aug path found



Bloom found

θ = least common ancestor of v,w

B = blossom

=

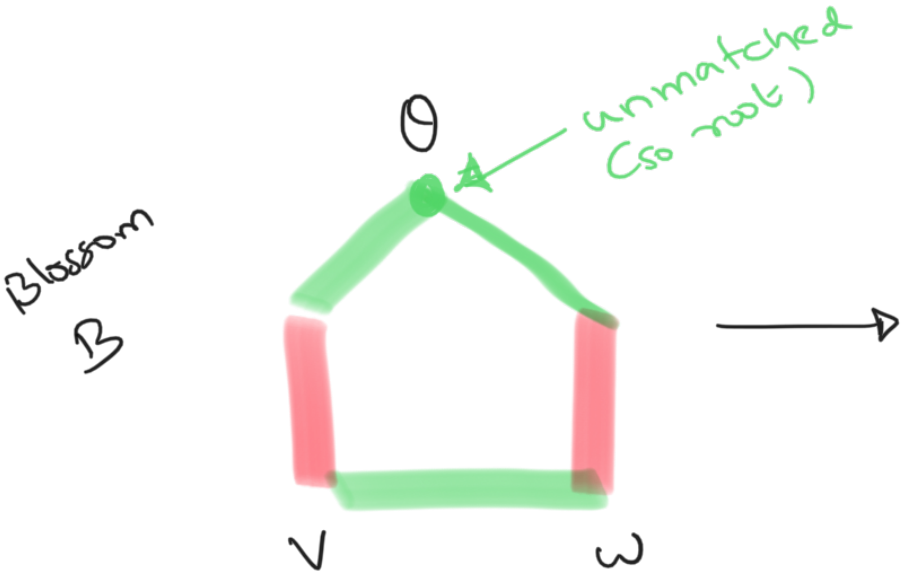
P' = Find any path in $(G/_B, M/_B)$

where $B \longrightarrow w \ldots$

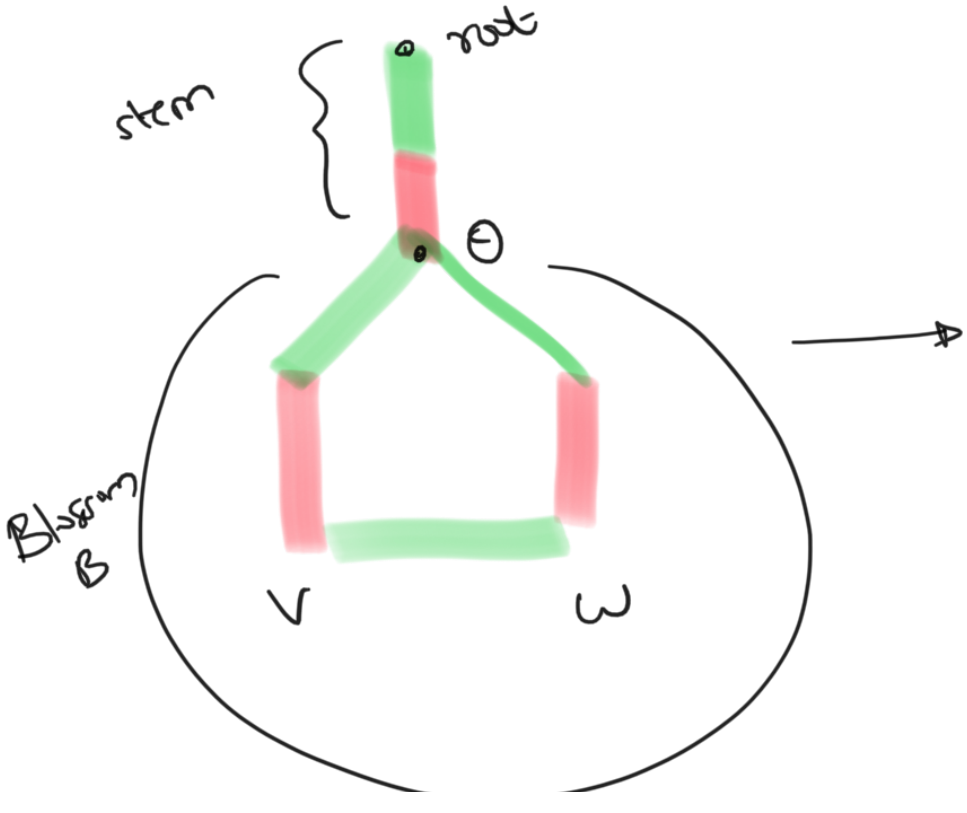
if $w \notin P'$, return P' (correct)

else $P = \text{lift}(P')$, return P

more details



contract to unmatched vertex b



contract to matched vertex b

→ if any path in $G/_B$ does not contain b
return itself

→ if it contains b.

b unmatched

b matched