How to find an augmenting path?

Construct an alternating tree.

1. **even nodes**
2. **odd nodes**

Case 4:
- \( y \) is already contained in \( T \) as an even vertex
- can't ignore \( y \)

The cycle \( w \leftrightarrow y \leftrightarrow x \leftrightarrow w \) is called a blossom.
- \( w \) is called the base of the blossom (even node!!!).
- The path \( u \leftrightarrow w \) is called the stem of the blossom.

---

Flowers and Blossoms

**Definition 1**

A flower in a graph \( G = (V,E) \) w.r.t. a matching \( M \) and a (free) root node \( r \), is a subgraph with two components:

- A stem is an even length alternating path that starts at the root node \( r \) and terminates at some node \( w \). We permit the possibility that \( r = w \) (empty stem).
- A blossom is an odd length alternating cycle that starts and terminates at the terminal node \( w \) of a stem and has no other node in common with the stem. \( w \) is called the base of the blossom.

**Properties:**

1. A stem spans \( 2\ell + 1 \) nodes and contains \( \ell \) matched edges for some integer \( \ell \geq 0 \).
2. A blossom spans \( 2k + 1 \) nodes and contains \( k \) matched edges for some integer \( k \geq 1 \). The matched edges match all nodes of the blossom except the base.
3. The base of a blossom is an even node (if the stem is part of an alternating tree starting at \( r \)).
Flowers and Blossoms

Properties:

4. Every node $x$ in the blossom (except its base) is reachable from the root (or from the base of the blossom) through two distinct alternating paths; one with even and one with odd length.

5. The even alternating path to $x$ terminates with a matched edge and the odd path with an unmatched edge.

Shrinking Blossoms

When during the alternating tree construction we discover a blossom $B$ we replace the graph $G$ by $G' = G/B$, which is obtained from $G$ by contracting the blossom $B$.

- Delete all vertices in $B$ (and its incident edges) from $G$.
- Add a new (pseudo-)vertex $b$. The new vertex $b$ is connected to all vertices in $V \setminus B$ that had at least one edge to a vertex from $B$.

- Edges of $T$ that connect a node $u$ not in $B$ to a node in $B$ become tree edges in $T'$ connecting $u$ to $b$.
- Matching edges (there is at most one) that connect a node $u$ not in $B$ to a node in $B$ become matching edges in $M'$.
- Nodes that are connected in $G$ to at least one node in $B$ become connected to $b$ in $G'$.
Shrinking Blossoms

- Edges of $T$ that connect a node $u$ not in $B$ to a node in $B$ become tree edges in $T'$ connecting $u$ to $b$.
- Matching edges (there is at most one) that connect a node $u$ not in $B$ to a node in $B$ become matching edges in $M'$.
- Nodes that are connected in $G$ to at least one node in $B$ become connected to $b$ in $G'$.

Example: Blossom Algorithm

Correctness

Assume that in $G$ we have a flower w.r.t. matching $M$. Let $r$ be the root, $B$ the blossom, and $w$ the base. Let graph $G' = G/B$ with pseudonode $b$. Let $M'$ be the matching in the contracted graph.

Lemma 2
If $G'$ contains an augmenting path $P'$ starting at $r$ (or the pseudo-node containing $r$) w.r.t. the matching $M'$ then $G$ contains an augmenting path starting at $r$ w.r.t. matching $M$.

Correctness

Proof.
If $P'$ does not contain $b$ it is also an augmenting path in $G$.

Case 1: non-empty stem
- Next suppose that the stem is non-empty.
Correctness

- After the expansion $\ell$ must be incident to some node in the blossom. Let this node be $k$.
- If $k \neq w$ there is an alternating path $P_2$ from $w$ to $k$ that ends in a matching edge.
- $P_1 \circ (i, w) \circ P_2 \circ (k, \ell) \circ P_3$ is an alternating path.
- If $k = w$ then $P_1 \circ (i, w) \circ (w, \ell) \circ P_3$ is an alternating path.

Proof.

Case 2: empty stem

- If the stem is empty then after expanding the blossom, $w = r$.

The path $r \circ P_2 \circ (k, \ell) \circ P_3$ is an alternating path.

Lemma 3

If $G$ contains an augmenting path $P$ from $r$ to $q$ w.r.t. matching $M$ then $G'$ contains an augmenting path from $r$ (or the pseudo-node containing $r$) to $q$ w.r.t. $M'$.

Proof.

- If $P$ does not contain a node from $B$ there is nothing to prove.
- We can assume that $r$ and $q$ are the only free nodes in $G$.

Case 1: empty stem

Let $i$ be the last node on the path $P$ that is part of the blossom. $P$ is of the form $P_1 \circ (i, j) \circ P_2$, for some node $j$ and $(i, j)$ is unmatched.

$(b, j) \circ P_2$ is an augmenting path in the contracted network.
Correctness

Illustration for Case 1:

![Graph Illustration]

Case 2: non-empty stem

Let $P_3$ be alternating path from $r$ to $w$; this exists because $r$ and $w$ are root and base of a blossom. Define $M_+ = M \oplus P_3$.

In $M_+$, $r$ is matched and $w$ is unmatched.

$G$ must contain an augmenting path w.r.t. matching $M_+$, since $M$ and $M_+$ have same cardinality.

This path must go between $w$ and $q$ as these are the only unmatched vertices w.r.t. $M_+$.

For $M'_+$, the blossom has an empty stem. Case 1 applies.

$G'$ has an augmenting path w.r.t. $M'_+$. It must also have an augmenting path w.r.t. $M'$, as both matchings have the same cardinality.

This path must go between $r$ and $q$.

Algorithm 24

\begin{algorithm}
\caption{search($r$, found)}
\begin{algorithmic}[1]
\STATE set $\bar{A}(i) \leftarrow A(i)$ for all nodes $i$
\STATE found $\leftarrow$ false
\STATE unlabel all nodes;
\STATE give an even label to $r$ and initialize list $\leftarrow \{r\}$
\WHILE{list $\neq \emptyset$}
\STATE delete a node $i$ from list
\STATE examine($i$, found)
\IF{found $= \text{true}$} \textbf{return} \ENDIF
\ENDWHILE
\end{algorithmic}
\end{algorithm}

Search for an augmenting path starting at $r$.

Algorithm 25

\begin{algorithm}
\caption{examine($i$, found)}
\begin{algorithmic}[1]
\FORALL{$j \in \bar{A}(i)$}
\IF{$j$ is even} contract($i,j$) and \textbf{return} \ENDIF
\IF{$j$ is unmatched}$q \leftarrow j$;\newline\quad pred($q$) $\leftarrow i$;\newline\quad found $\leftarrow$ true;\newline\quad \textbf{return} \ENDIF
\IF{$j$ is matched and unlabeled}$\quad$ pred($j$) $\leftarrow i$;\newline\quad$\quad$ pred(mate($j$)) $\leftarrow j$;\newline\quad$\quad$ add mate($j$) to list
\ENDFOR
\end{algorithmic}
\end{algorithm}

Examine the neighbours of a node $i$.
Algorithm 26 contract\((i, j)\)
1: trace pred-indices of \(i\) and \(j\) to identify a blossom \(B\)
2: create new node \(b\) and set \(\bar{A}(b) = \bigcup_{x \in B} \bar{A}(x)\)
3: label \(b\) even and add to list
4: update \(\bar{A}(j) = \bar{A}(j) \cup \{b\}\) for each \(j \in \bar{A}(b)\)
5: form a circular double linked list of nodes in \(B\)
6: delete nodes in \(B\) from the graph

Identify all neighbours of \(b\).
Time: \(\mathcal{O}(m)\) (why?)

Contract blossom identified by nodes \(i\) and \(j\)

Get all nodes of the blossom.
Time: \(\mathcal{O}(m)\)
**Algorithm 26 contract** \((i, j)\)

1. trace pred-indices of \(i\) and \(j\) to identify a blossom \(B\)
2. create new node \(b\) and set \(\bar{A}(b) = \bigcup_{x \in B} \bar{A}(x)\)
3. label \(b\) even and add to list
4. update \(\bar{A}(j) = \bar{A}(j) \cup \{b\}\) for each \(j \in \bar{A}(b)\)
5. form a circular double linked list of nodes in \(B\)
6. delete nodes in \(B\) from the graph

Every node that was adjacent to a node in \(B\) is now adjacent to \(b\).

**Algorithm 26 contract** \((i, j)\)

1. trace pred-indices of \(i\) and \(j\) to identify a blossom \(B\)
2. create new node \(b\) and set \(\bar{A}(b) = \bigcup_{x \in B} \bar{A}(x)\)
3. label \(b\) even and add to list
4. update \(\bar{A}(j) = \bar{A}(j) \cup \{b\}\) for each \(j \in \bar{A}(b)\)
5. form a circular double linked list of nodes in \(B\)
6. delete nodes in \(B\) from the graph

Only for making a blossom expansion easier.

---

**Analysis**

- A contraction operation can be performed in time \(O(m)\). Note, that any graph created will have at most \(m\) edges.
- The time between two contraction-operation is basically a BFS/DFS on a graph. Hence takes time \(O(m)\).
- There are at most \(n\) contractions as each contraction reduces the number of vertices.
- The expansion can trivially be done in the same time as needed for all contractions.
- An augmentation requires time \(O(n)\). There are at most \(n\) of them.
- In total the running time is at most

\[
n \cdot (O(mn) + O(n)) = O(mn^2)
\]

Only delete links from nodes not in \(B\) to \(B\). When expanding the blossom again we can recreate these links in time \(O(m)\).
Example: Blossom Algorithm

Animation of Blossom Shrinking algorithm is only available in the lecture version of the slides.