Overview: Shortest Augmenting Paths

**Lemma 1**

*The length of the shortest augmenting path never decreases.*

**Lemma 2**

*After at most $\mathcal{O}(m)$ augmentations, the length of the shortest augmenting path strictly increases.*
These two lemmas give the following theorem:

**Theorem 3**

The shortest augmenting path algorithm performs at most $O(mn)$ augmentations. This gives a running time of $O(m^2n)$.

**Proof.**

- We can find the shortest augmenting paths in time $O(m)$ via BFS.
- $O(m)$ augmentations for paths of exactly $k < n$ edges.
Define the level $\ell(v)$ of a node as the length of the shortest $s-v$ path in $G_f$.

Let $L_G$ denote the subgraph of the residual graph $G_f$ that contains only those edges $(u, v)$ with $\ell(v) = \ell(u) + 1$.

A path $P$ is a shortest $s-u$ path in $G_f$ if it is a $s-u$ path in $L_G$. 
In the following we assume that the residual graph $G_f$ does not contain zero capacity edges.

This means, we construct it in the usual sense and then delete edges of zero capacity.
Shortest Augmenting Path

First Lemma:
The length of the shortest augmenting path never decreases.

After an augmentation $G_f$ changes as follows:

- Bottleneck edges on the chosen path are deleted.
- Back edges are added to all edges that don’t have back edges so far.

These changes cannot decrease the distance between $s$ and $t$. 
**Shortest Augmenting Path**

**Second Lemma:** After at most $m$ augmentations the length of the shortest augmenting path strictly increases.

Let $E_L$ denote the set of edges in graph $L_G$ at the beginning of a round when the distance between $s$ and $t$ is $k$.

An $s$-$t$ path in $G_f$ that uses edges not in $E_L$ has length larger than $k$, even when considering edges added to $G_f$ during the round.

In each augmentation one edge is deleted from $E_L$. 

![Graph diagram]
Shortest Augmenting Paths

Theorem 4
The shortest augmenting path algorithm performs at most $O(mn)$ augmentations. Each augmentation can be performed in time $O(m)$.

Theorem 5 (without proof)
There exist networks with $m = \Theta(n^2)$ that require $O(mn)$ augmentations, when we restrict ourselves to only augment along shortest augmenting paths.

Note:
There always exists a set of $m$ augmentations that gives a maximum flow (why?).
When sticking to shortest augmenting paths we cannot improve (asymptotically) on the number of augmentations.

However, we can improve the running time to $O(mn^2)$ by improving the running time for finding an augmenting path (currently we assume $O(m)$ per augmentation for this).
Shortest Augmenting Paths

We maintain a subset $E_L$ of the edges of $G_f$ with the guarantee that a shortest $s$-$t$ path using only edges from $E_L$ is a shortest augmenting path.

With each augmentation some edges are deleted from $E_L$.

When $E_L$ does not contain an $s$-$t$ path anymore the distance between $s$ and $t$ strictly increases.

Note that $E_L$ is not the set of edges of the level graph but a subset of level-graph edges.
Suppose that the initial distance between $s$ and $t$ in $G_f$ is $k$. 

$E_L$ is initialized as the level graph $L_G$.

Perform a **DFS search** to find a path from $s$ to $t$ using edges from $E_L$.

Either you find $t$ after at most $n$ steps, or you end at a node $v$ that does not have any outgoing edges.

You can delete incoming edges of $v$ from $E_L$. 


Let a phase of the algorithm be defined by the time between two augmentations during which the distance between \( s \) and \( t \) strictly increases.

Initializing \( E_L \) for the phase takes time \( \mathcal{O}(m) \).

The total cost for searching for augmenting paths during a phase is at most \( \mathcal{O}(mn) \), since every search (successful (i.e., reaching \( t \)) or unsuccessful) decreases the number of edges in \( E_L \) and takes time \( \mathcal{O}(n) \).

The total cost for performing an augmentation during a phase is only \( \mathcal{O}(n) \). For every edge in the augmenting path one has to update the residual graph \( G_f \) and has to check whether the edge is still in \( E_L \) for the next search.

There are at most \( n \) phases. Hence, total cost is \( \mathcal{O}(mn^2) \).