# Combinatorial Optimization

## References


- Christos H. Papadimitriou, Kenneth Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall 1982.
- Robert Endre Tarjan, Data Structures and Network Algorithms, SIAM 1983.
- M. Gondran, M. Minoux, Graphs and Algorithms, Wiley & Sons 1984.
- L. Lovász, M.D. Plummer, Matching Theory, North-Holland 1986.
- G.L. Nemhauser, A.H.G. Rinnooy Kan, M.J. Todd, Optimization, Handbooks in Operations Research and Management Science Vol. 1, North-Holland 1989
- Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall 1993.
- R.L. Graham, M. Grötschel, L. Lovász, eds. Handbook of Combinatorics Vol I, Vol II, Elsevier 1995.
- Alexander Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer 2003.

- Jørgen Bang-Jensen, Gregory Gutin, Digraphs: Theory, Algorithms and Applications, 2nd Ed., Springer 2009.
- Mokhtar S. Bazaraa, John J. Jarvis, Hanif D. Sherali, Linear Programming and Network Flows, 4th Ed., Wiley & Sons 2010.

## Contents

# 1. Matching algorithms

We continue the lecture "Pearls of Algorithms, Part 1" given last semester. It is assumed that the lecture notes

" Pearls of Algorithms
1. Bipartite matching and network flows"

are well known.

## 1.1 Nonbipartite graphs

### 1.1.1 Reduction to a reachability problem

Review of the bipartite case:

Given $G = (A, B, E)$, matching $M \subseteq E$

$\rightsquigarrow$

construction of a directed bipartite graph

$G_M = (A', B', E_M)$ where

$A' = A \cup \{s\}$, $B' = B \cup \{t\}$, $s, t \notin A \cup B$, $s \neq t$

and

$$
\begin{aligned}
E_M = &\{(u,v) \mid (u,v) \in M \text{ and } u \in A, v \in B\} \\
&\dot\cup \{(x,y) \mid (x,y) \in E \setminus M \text{ and } x \in B, y \in A\} \\
&\cup \{(s,b) \mid b \in B \text{ M-free}\} \\
&\cup \{(a,t) \mid a \in A \text{ M-free}\}
\end{aligned}
$$

Property

$\exists$ M-augmenting path in G $\Longleftrightarrow$ $\exists$ simple path from s to t in $G_M$

$\leadsto$ Apply DFS with start node s to $G_M$

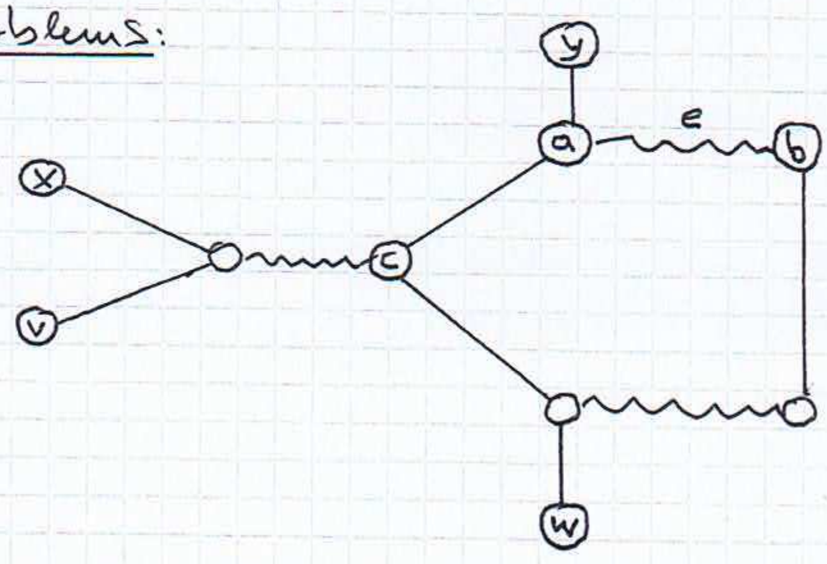
General case:

Let $G = (V, E)$ be an undirected graph and $M \subseteq E$ be a matching.

$$V_M := \{ x \in V \mid x \text{ is } M\text{-free} \}$$

Idea: Analogously to the bipartite case, construct a directed graph $G_M$.

Problems:




- M-augmenting path from x to y enters the edge e in b and leaves the edge e from a
- M-augmenting path from v to w enters e in a and leaves e from b.

$\rightsquigarrow$

- A priori, we cannot divide the set of nodes $V$ into two sets $A$ and $B$ such that an $M$-augmenting path exists in $G$ iff there exists an $M$-augmenting path using alternately nodes from $A$ and from $B$.

<u>Idea</u>

Double each node $v \in V$ and add one of these nodes to $A$ and the other to $B$. Direct the edges in $M$ from $A$ to $B$ and the edges in $E \setminus M$ from $B$ to $A$.

$\rightsquigarrow$

$$G_M = (V', E_M) \text{ where}$$

$V' = \{[v,A], [v,B] \mid v \in V\} \cup \{s,t\}$ , $s,t \notin V$, $s \neq t$

$$\begin{aligned} E_M = \; & \{([v,A],[w,B]), ([w,A],[v,B]) \mid (v,w) \in M\} \\ & \cup \{([x,B],[y,A]), ([y,B],[x,A]) \mid (x,y) \in E \setminus M\} \\ & \cup \{(s,[v,B]), ([v,A],t) \mid v \in V_M\} \end{aligned}$$

Since the distinct nodes $[v,A]$ and $[v,B]$ in $V'$ correspond to the same node $v \in V$, it does not suffice to construct a simple path from $s$ to $t$ in $G_M$ for finding an $M$-augmenting path in $G$.

$\rightsquigarrow$

A path $P$ in $G_M$ is <u>strongly simple</u> if

a) $P$ is simple, and
b) $\forall [v,A] \in V'$: $[v,A] \in P \Rightarrow [v,B] \notin P$.

Now we can formulate the reachability problem in $G_M$ which is equivalent to the problem of finding an M-augmenting path in G.

Theorem 1.1

Let $G = (V, E)$ be an undirected graph, $M \subseteq E$ be a matching, and $G_M = (V', E_M)$ be defined as above. Then there exists an M-augmenting path in G if and only if there exists a strongly simple path from s to t in $G_M$.

Proof:

„$\Leftarrow$"

Let

$$P = s, [v_1, B], [v_2, A], [v_3, B], \ldots, [v_{k-1}, B], [v_k, A], t$$

be a strongly simple path in $G_M$. Then

$$v_i \neq v_j \;,\; 1 \leq i < j \leq k \quad \text{and} \quad v_1, v_k \in V_M .$$

$\Rightarrow$

$$P' = v_1, v_2, \ldots, v_k$$

is an M-augmenting path in G.

„$\Rightarrow$"

Let

$$Q = w_1, w_2, \ldots, w_{e-1}, w_e$$

be an M-augmenting path in G. Then

$w_i \neq w_j$, $1 \leq i < j \leq \ell$ and $w_1, w_\ell \in V_M$.

$\Longrightarrow$ by construction

$$Q' = s, [w_1, B], [w_2, A], \ldots, [w_{\ell-1}, B], [w_\ell, A], t$$

is a strongly simple path in $G_M$.

## 1.1.2 The solution of the reachability problem

Goal:

Solution of the reachability problem in $G_M$.

DFS finds simple paths in a directed graph. Hence, we cannot use DFS directly for the solution of the reachability problem in $G_M$.

Idea:

Modify the usual DFS such that the modified depth-first search (MDFS) finds precisely the strongly simple paths in $G_M$.

Let $[v, \bar{A}] = [v, B]$ and $[v, \bar{B}] = [v, A]$.

Remember that a DFS partitions the edges of the graph into four categories. Similary, the edges of $G_M$ are partitioned into five categories by a MDFS of $G_M$:

1. Tree edges, which are edges leading to new nodes $[v, X]$, $X \in \{A, B\}$, for which $[v, \bar{X}]$ is not a predecessor during the search.

2. Weak back edges, which are edges leading to new nodes $[v, A]$, for which $[v, B]$ is a predecessor during the search.

3. Back edges, which go from descendants to ancestors during the search

4. Forward edges, which go from ancestors to proper descendants but are not tree edges.

5. Cross edges, which go between nodes that are neither ancestors nor descendants of one another during the search.

Like DFS, MDFS uses a stack $K$ for the organisation of the search. Analogously to DFS, the MDFS-stack $K$ defines a tree, the MDFS-tree $T$.

Notation:

TOP($K$) last node added to $K$.

In each step, MDFS considers an edge $(\mathrm{TOP}(K), [w, Y])$ which was not considered previously. Let

$$e = ([v, X], [w, \bar{X}])$$

be the edge under consideration. We distinguish two cases:

1. $X = A$ ; i.e., $(v, w) \in M$ (tree edge)

2. $X = B$ ; i.e., $(v, w) \in E \setminus M$

   2.1 $[w, A] \in K$ (back edge)

   2.2. $[w, A] \notin K$ but $[w, B] \in K$
   i) $[w, A]$ has been in $K$ previously (cross edge)
   ii) $[w, A]$ has not been in $K$ previously (weak back edge)

   2.3 $[w, A] \notin K$ and $[w, B] \notin K$
   i) $[w, A]$ has been in $K$ previously (forward or cross edge)
   ii) $[w, A]$ has not been in $K$ previously (tree edge)

MDFS differs from DFS only in cases 2.2.ii and 2.3.i. Next, we will discuss both of these cases.

<u>Case 2.2.ii</u>

Since $[w, A]$ has not been in $K$ before, DFS would perform the operation PUSH($[w, A]$). Since $[w, B] \in K$ and MDFS should only construct strongly simple paths in $G_M$, MDFS does not perform the operation PUSH($[w, A]$).

Note that the path

$$P = s, P_1, [w, B], P_2, [v, B]$$

defined by the MDFS-stack $K$ is strongly simple. Hence, for each node $[x, A]$ on $P_2$, the following is fulfilled:

Let

$$P_2 = P_{21}, [x,A], P_{22} \quad \text{and} \quad Q = [x,A], P_{22}, [v,B], [w,A]$$

Then the path $Q$ is strongly simple.

We say that MDFS has <u>found</u> the strongly simple path $Q$ from $[x,A]$ to $[w,A]$.

Since $Q$ is above the node $[w,B]$, after the performance of the operation POP$([w,B])$, no node on $Q$ is in the MDFS-stack $K$. Moreover, as we shall prove later, for all nodes $[z,X]$ on $P_2$ the operations PUSH$([z,X])$, POP$([z,X])$, PUSH$([z,\overline{X}])$ and POP$([z,\overline{X}]$ are performed before POP$([w,B])$.

<u>Case 2.3.i</u>

Since $[w,A]$ has been in $K$ before, DFS would perform no PUSH-operation. But the different treatment of Case 2.2.ii can cause the following situation:

- MDFS has found a strongly simple path $Q = [w,A], Q', [u,A]$ from the node $[w,A]$ to a node $[u,A]$ but at that moment, the node $[u,B]$ was below $[w,\overline{A}]$ in the MDFS-stack $K$ such that the operation PUSH$([u,A])$ has not been performed. But now, $[u,B] \notin K$.

As we shall prove later, the paths $P$ from $s$ to $[v,B]$

and $Q$ from $[w,A]$ to $[u,A]$ are strongly disjoint, i.e., there is no $[r,X]$ on $P$, $X \in \{A,B\}$ such that $\{[r,A],[r,B]\} \cap Q \neq \emptyset$. Since MDFS has "found" a strongly simple path $P,Q$ from $s$ to $[u,A]$, MDFS now performs the operation PUSH($[u,A]$).

Note that with respect to depth-first search, the DFS-stack always contains the current search path. With respect to the modified depth-first search, the situation is different. In Case 2.3.i, the node $[u,A]$ is pushed. But to obtain a current search path, between the nodes $[v,B]$ and $[u,A]$, we have to insert any strongly simple path $[w,A], Q', [u,A]$ which has been found by MDFS. Since we do not want to forgot the information about the first node on the path which we add between the nodes $[v,B]$ and $[u,A]$, we create the artificial tree edge $([v,B],[u,A])_{[w,A]}$. Such an edge is called extensible edge.

It is possible that there exists various such paths $Q'$. Hence, after the performance of PUSH($[u,A]$), the number of corresponding current search paths can increase.

Always if we consider one current search path we mean that we can take an arbitrary corresponding current search path.

If we add to the current MDFS-tree $T$ all forward, back, cross and weak back edges and replace every extensible edge $([v,B],[u,A])_{[w,A]}$ by all strongly simple paths

$$Q = [v,B],[w,A],Q',[u,A]$$

such that all edges on $Q$ are contained in the extended current MDFS-tree $T$ then we obtain the <u>expanded current MDFS-tree</u> $T_{exp}$.

We say that MDFS has <u>constructed</u> a path $P$ if the MDFS-stack $K$ contains the path $P$ where each extensible edge $([v,B],[u,A])_{[w,A]}$ in $K$ is replaced by one of the strongly simple paths $Q=[v,B],[w,A],Q',[u,A]$ which replace this extensible edge in $T_{exp}$.

We say that MDFS has <u>formed</u> a strongly simple path $P$ if $T_{exp}$ contains $P$. We say that MDFS has <u>found</u> a strongly simple path $P',[v,B],[w,A]$ if the path $P',[v,B]$ is formed by MDFS and the edge $([v,B],[w,A])$ is a considered weak back edge.

Next we shall describe MDFS more in detail. We have to solve the following problem:

How to find the node $[u,A]$ in Case 2.3.i?

For the solution of this problem, we assume that MDFS is organized such that for all nodes $[w, A] \in V'$, the following holds true:

After performing the operation POP($[w,A]$), MDFS has always computed a set $L_{[w,A]}$ of nodes such that $L_{[w,A]}$ contains exactly those nodes $[u,A] \in V'$ satisfying the requirements that

1. MDFS has found a strongly simple path $P = [w,A], Q, [u,A]$.
2. PUSH($[u,A]$) has never been performed, and
3. POP($[u,B]$) has been performed.

Note that $[u,B] \notin Q$. Before the performance of POP($[w,A]$), we fix $L_{[w,A]} := \emptyset$.

In the description of MDFS we assume for all $[w,A] \in V'$ that $L_{[w,A]}$ is computed correctly. As we will prove later, always $|L_{[w,A]}| \leq 1$. For $[v,X] \in V'$, $N[v,X]$ denotes the adjacency list of $[v,X]$.

<u>Algorithm</u> MDFS

<u>Input</u>: $G_M = (V', E_M)$

<u>Output</u>: A strongly simple path $P$ from $s$ to $t$, if such a path exists.

Method:

PUSH(s);
while $K \neq \emptyset$ and no path from s to t is constructed
do
SEARCH
od.

SEARCH is a call of the following procedure.

```
procedure SEARCH
if TOP(K) = t
  then
    reconstruct a strongly simple path
    P from s to t which has been
    constructed by the algorithm
  else
    mark TOP(K) "pushed";
    for all nodes [w, Y] ∈ N[TOP(K)]
      do
        if Y = B
(Case 1)  then
            PUSH([w, B]);
            SEARCH
(Case 2)  else
            if [w, A] ∈ K
(Case 2.1)    then
```

no PUSH-operation
else
if $[w,B] \in K$
(Case 2.2) then
no PUSH-operation
(Case 2.3) else
if $[w,A]$ is marked "pushe[d]"
(Case 2.3.i) then
while $L_{[w,A]} \neq \emptyset$
do
choose any $[u,A] \in L_{[w,A]}$
PUSH($[u,A]$);
SEARCH
od
(Case 2.3.ii) else
PUSH($[w,A]$);
SEARCH
fi
fi
fi
fi
od
POP;
POP
fi.

## 1.1.3 The correctness proof of MDFS

This proof is inspired by the correctness proof of DFS. But in contrast to DFS, the proof is difficult.

The difficulties come from the fact that the MDFS-stack does not contain the whole current search path and the decisions taken by the algorithm only depend on the content of the current stack. Hence, the proof that the algorithm constructs only strongly simple paths is involved.

First, we shall prove some lemmas. The first lemma implies that the first PUSH-operation which destroys the property "strongly simple" must push a node with second component A.

Lemma 1.1

As long as MDFS constructs only strongly simple paths, the following holds true: After the operation PUSH([u, A]) where $v$ is not M-free, the operation PUSH([w, B]) where $([v, A], [w, B]) \in E_M$ always follows without destroying the property "strongly simple".

Proof:

After the performance of the operation PUSH([v, A]) MDFS always consider the unique edge $([v, A], [w, B]) \in E_M$ and performs the operation PUSH([w, B]). If this operation destroys the property "strongly simple", then [w, A] and hence, [v, B] would be on a current search path. But then the operation PUSH([v, A]) would have destroyed the property "strongly simple", a contradiction.

∎

The next lemma shows that MDFS constructs a path from $s$ to a node $[x, A]$ if in a specific situation a strongly simple path from $s$ to this node exists.

Lemma 1.2

Let $[u, B] \in V'$ be a node for which MDFS performs the operation PUSH($[u,B]$). Furthermore, at the moment when POP($[u,B]$) is performed by MDFS, only strongly simple paths have been constructed by MDFS. Let $[x,A] \in V'$ such that at the moment when PUSH($[u,B]$) is performed, there is a strongly simple path

$$P = [u,B], [v,A], Q, [x,A]$$

with $[z,X], [z,\bar{X}] \notin K$ for all $[z,X] \in P$. Then PUSH($[x,A]$) has been performed before POP($[u,B]$).

Remark:

Lemma 1.2 implies that either PUSH($[x,A]$) and POP($[x,A]$) have been performed before the performance of PUSH($[u,B]$), or both operations have been performed between the operations PUSH($[u,B]$) and POP($[u,B]$).

Proof:

Let

$$P = [u,B],[v,A][v',B], Q', [x,A]$$

be such a path of shortest length for which PUSH($[x,A]$) has not been performed before POP($[u,B]$).

It is clear that the edge $e = ([u,B],[v,A])$ has been considered before the performance of POP($[u,B]$). Since at the moment when $e$ is considered by assumption $[v,A] \notin K$ and $[v,B] \notin K$, MDFS is in Case 2.3.

If the operation PUSH($[v,A]$) is performed according to this consideration of edge $e$, then by the assumption that $P$ is a shortest path such that the assertion is not fulfilled, PUSH($[x,A]$) has been performed before POP($[v',B]$), and hence, before POP($[u,B]$).

Hence, MDFS is in Case 2.3.i and performs the corresponding while-statement. Consider the moment when MDFS finishs this while-statement; i.e., $L_{[v,A]} = \emptyset$.

Let $[z,A] \in P$ be the first node on $P$ for which PUSH($[z,A]$) has not been performed. Since $[x,A]$ has this property, the node $[z,A]$ exists. Let

$$P = \underbrace{[u,B],[v,A], Q_1, [y,B]}_{P_1}, [z,A], Q_2, [x,A]$$

By construction, each node on $P_1$ is pushed and each edge on $P_1$ considered.

$\Rightarrow$

All these nodes and edges are in $T_{exp}$ such that $P_1$ is formed by MDFS. Furthermore, the edge $([y,B],[z,A])$ is a considered weak back edge.

$\Rightarrow$

1. MDFS has found the path $P_1, [z,A]$

By assumption

2. PUSH($[z,A]$) has never been performed.

Since $[z,B] \notin K$ when PUSH($[u,B]$) is performed there holds:

3. POP($[z,B]$) has been performed.

Hence, $[z,A] \in L_{[v,A]}$ and hence, $L_{[v,A]} \neq \emptyset$. But this contradicts $L_{[v,A]} = \emptyset$. such that the lemma is proved.

∎

For $v \in V'$, we denote

$$r(w) := \begin{cases} [v,\overline{X}] & \text{if } w = [v,X] \\ t & \text{if } w = s \\ s & \text{if } w = t \end{cases}$$

Let $S = w_1, w_2, \ldots, w_k$ be a path in $G_M$. The <u>backpath</u> $r(S)$ of $S$ is defined by

$$r(S) = r(w_k), r(w_{k-1}), \ldots, r(w_1).$$

<u>Lemma 1.3</u>

Let $[u,B] \in V'$ be a node for which MDFS performs the operation PUSH($[u,B]$). Furthermore, at the moment when POP($[u,B]$) is performed by MDFS, only strongly simple paths have been constructed by MDFS. If there exists a strongly simple path $P = [v,A], Q, [w,B]$ such that at the moment when PUSH($[u,B]$) is performed, $[z,X], [z,\overline{X}] \notin$, for all $[z,X] \in P$, and $([u,B],[v,A]), ([w,B],[u,A]) \in E_M$, then for all $[z,X] \in P$, the operations PUSH($[z,X]$) and PUSH($[z,\overline{X}]$) have been performed before the operation POP($[u,B]$).

<u>Proof</u>:

For the nodes $[z,X] \in P$ consider the path $[u,B], P$ and apply Lemma 1.2. For the nodes $[z,\overline{X}]$ consider the path $[u,B], r(P)$ and apply Lemma 1.2.

Note that by the definition of $L_{[u,A]}$

$|L_{[u,A]}| > 0 \implies$ PUSH($[u,A]$) and POP($[u,A]$) have been performed.

Lemma 1.4

MDFS maintains the following invariants:

1. MDFS constructs only strongly simple paths.
2. $|L_{[w,A]}| \leq 1$, for all $[w,A] \in V'$.
3. Assume that the algorithm performs the assignment $L_{[w,A]} := [u,A]$. Then after the performance of PUSH($[u,A]$), always $L_{[w,A]} = L_{[u,A]}$.

Remark:

Invariant 2 and Invariant 3 are not needed for the correctness proof of MDFS. But we shall need these invariants for the efficient implementation of the algorithm. Moreover, the proof of Invariant 1 is easier if we prove all invariants simultaneously.

Proof:

Consider the first situation in which one of the three invariants is not maintained. Three cases are to be considered.

Case 1: Invariant 1 is not maintained.

Only a PUSH-operation can destroy the propert[y] "strongly simple". Note that a PUSH-operation cannot affect Invariant 2 or Invariant 3.

Lemma 1.1 implies that this PUSH-operation occurs during the consideration of an edge $e = ([v,B],[w,A])$. Then $e$ corresponds to edge $(v,w) \in E \setminus M$.

If $[w,A]$ is not marked "pushed", then Case 2.3.ii of MDFS applies, and PUSH($[w,A]$) is performed. Note that $[w,B] \notin K$. Hence, the only possible situation in which this PUSH-operation destroys the property "strongly simple" is the following:

On a current search path there is a subpath Q which is caused by an application of Case 2.3.i of MDFS such that $[w,B] \in Q$.

$\Rightarrow$

$\exists\ [u,A] \in V'$ such that the addition of Q to this current search path is caused by the operation PUSH($[u,A]$).

Construction $\Rightarrow$

The assumptions of Lemma 1.3 are fulfilled with respect to $[u,B]$ and $[w,B] \in P$.

Hence, by Lemma 1.3, PUSH([w,A]) has been performed <u>before</u> POP([u,B]), and hence, <u>before</u> PUSH([u,A]), a contradiction

$\Longrightarrow$

[w,A] is marked "pushed" such that Case 2.3.i of MDFS applies.

Note that by Invariant 2, $|L_{[w,A]}| \leq 1$. We thus write $L_{[w,A]} = [u,A]$ instead of $L_{[w,A]} = \{[u,A]\}$.

$\Longrightarrow$

For the node $[u,A] = L_{[w,A]}$, the operation PUSH([u,A]) is performed.

$\Longrightarrow$

MDFS extends the current search paths by a path [w,A], Q, [u,A]. but only [u,A] is pushed.

By the definition of $L_{[w,A]}$ and by Lemma 1.3 the operations PUSH([z,X]), POP([z,X]), PUSH([z,$\bar{X}$]) and POP([z,$\bar{X}$]) have been performed for all $[z,X] \in Q$. such that none of these nodes is in the current MDFS-stack K.

$\Longrightarrow$

The only possible situation in which PUSH([u,A]) destroys the property "strongly simple" is the following:

There is a node $[p,X] \in [w,A], Q, [u,A]$ and a subpath $Q'$ of a current search path which is caused by an application of Case 2.3.i such that $[p,X] \in Q'$ or $[p,\bar{X}] \in Q'$.

Since one end node of an edge in the current matching uniquely determines the other end node, we can choose $[p,X]$ such that $[p,A] \in Q'$.

Consider node $[u',A] \in K$ with $\text{PUSH}([u',A])$ is the operation which adds the subpath $Q'$ to this current search path.

Definition of $L_{[p,A]}$, Lemma 1.3 and Invariant 2

$\Longrightarrow$

Before the performance of $\text{PUSH}([u',A])$ there holds

$$L_{[p,A]} = [u',A].$$

Hence, by Invariant 3, after the performance of $\text{PUSH}([u',A])$, always

$$L_{[p,A]} = L_{[u',A]}.$$

By the choice of $[p,A]$, $L_{[p,A]} = [u,A]$, and hence, $L_{[u',A]} = [u,A]$ in the situation under consideration.

$\Longrightarrow$

$\text{POP}([u',A])$ is performed; i.e., $[u',A] \notin K$, a contradiction.

<u>Case 2:</u> Invariant 2 is not maintained

Then there is $[w, A]$, $[p_1, A]$, $[p_2, A] \in V'$ with the property that

$$L_{[w,A]} = \{[p_1, A]\} \text{ before the performance of } \mathrm{POP}([p_2, B])$$

and

$$L_{[w,A]} = \{[p_1, A], [p_2, A]\}$$

after the performance of $\mathrm{POP}([p_2, B])$.

Hence, MDFS has found a path

$$P_1 = [p_1, B], Q, [p_1, A] \text{ with } [w, A] \in Q$$

and found a path

$$P_2 = [p_2, B], Q', [p_2, A] \text{ with } [w, A] \in Q'.$$

If MDFS has found the path $P_2$ after the performance of $\mathrm{POP}([p_1, B])$, then $[w, A]$ can only be added to $Q'$ in the following way:

An operation $\mathrm{PUSH}([u, A])$, caused by an application of Case 2.3.i with respect to a node $[v, A]$ (i.e., $[u, A] \in L_{[v,A]}$) is performed such that the current search path is extended by a path

$$[v, A], \tilde{Q}, [u, A] \text{ with } [w, A] \in \tilde{Q}.$$

$\Longrightarrow$ $[u, A] \in L_{[w,A]}$ before the performance of PUSH($[u,A]$)

PUSH($[u,A]$) is performed after POP($[p_1, B]$) since $P_2$ is found after POP($[p_1, B]$).

$\Longrightarrow$

$[u, A], [p_1, A] \in L_{[w,A]}$

between the performance of these two operatio

This contradicts the assumption that we consider the situation in which invariant 2 is not maintained for the first time.

Hence, MDFS has found the path $P_2$ before the performance of POP($[p_1, B]$).

Note that $[p_1, B] \notin Q'$. Otherwise, by Lemma 1.3, PUSH($[p_1, A]$) is performed before POP($[p_2, B$ and hence, $[p_1, A] \notin L_{[w,A]}$ after POP($[p_2, B]$).

Let $[r, A]$ be the first node on $Q'$ such that $[r, A] \in Q$ or $[r, B] \in Q$. Since node $[w, A]$ has this property, node $[r, A]$ exists. Let

$$Q' = Q'_1, [r, A], Q'_2$$

and

$$Q = \begin{cases} Q_1, [r,A], Q_2 & \text{if } [r,A] \in Q \\ Q_1, [r,B], Q_2 & \text{if } [r,B] \in Q \end{cases}$$

Consider the path

$$R = \begin{cases} Q_1', [r,A], Q_2, [p_1,A] & \text{if } [r,A] \in Q \\ Q_1', [r,A], \tau(Q_1), [p_1,A] & \text{if } [r,B] \in Q \end{cases}$$

Then Lemma 1.2 applies with respect to $[p_2,B]$, $[p_1,A]$ and the strongly simple path $R$.

$\Rightarrow$

PUSH($[p_1,A]$) is performed before POP($[p_2,B]$)

$\Rightarrow$

$[p_1,A] \notin L_{[w,A]}$ after POP($[p_2,B]$),

a contradiction.

<u>Case 3</u>: Invariant 3 is not maintained.

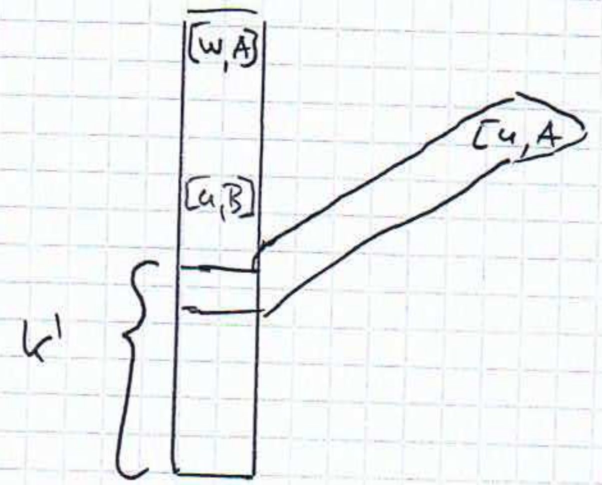
After the performance of PUSH ($[u,A]$), there holds $L_{[w,A]} = L_{[u,A]} = \emptyset$. We shall prove that $L_{[w,A]} = L_{[u,A]}$ after the next POP-operation which changes $L_{[w,A]}$ or $L_{[u,A]}$. Then, the assertion follows because of Invariant 2 and the transitivity of the relation $=$.

Let POP($[p,B]$) be the next POP-operation which enlarges $L_{[w,A]}$ or $L_{[u,A]}$.

$K_{[w,A]}$ denotes the current MDFS-stack, directly after the performance of PUSH($[w,A]$).

Let

$$k' = k_{[w,A]} \cap k_{[u,A]}.$$




According to the location of $[p,B]$ with respect to $k_{[w,A]}$ and to $k_{[u,A]}$, we distinguish three cases:

1) Construction $\Rightarrow$

$$[p,B] \notin k_{[w,A]} \setminus k'.$$

Otherwise, POP($[p,B]$) would be performed before PUSH($[u,A]$).

2) Assume that $[p,B] \in k_{[u,A]} \setminus k'$.

Let $[q,B]$ be the first node in $k_{[w,A]} \setminus k'$ such that $[q,A] \in k_{[u,A]} \setminus k_{[p,B]}$.

Node $[q,B]$ exists since $[u,B]$ has the property that $[u,B] \in k_{[w,A]} \setminus k'$.

Consider the back path of the path from node $[p,B]$ to node $[q,A]$.

This backpath implies that $[q,B]$ and $[p,A]$ fulfill the assumptions of Lemma 1.2.

$\Rightarrow$ PUSH($[p,A]$) occurs before POP($[q,B]$).

Since $[q,B] \in K_{[w,A]} \setminus K'$, the operation PUSH($[p,A]$) is also performed before POP($[p,B]$)

$\Rightarrow$ POP($[p,B]$) can enlarge neither $L_{[w,A]}$ nor $L_{[u,A]}$

3) It remains the case $[p,B] \in K'$!

Let $[q,B] \in K'$ be the node nearest to the top of $K'$ for which PUSH($[q,A]$) has not be performed at the moment when MDFS performs PUSH($[u,A]$)

Since $[p,B]$ has this property, $[q,B]$ exists.

By consideration of the backpath of the path from $[q,B]$ to $[u,B]$, it is easy to prove that MDFS finds a path from $[u,A]$ to $[q,A]$ not containing $[q,B]$. exercise

Hence, $L_{[u,A]} = [q,A]$ after the performance of POP($[q,B]$), and hence, $[q,B] = [p,B]$.

Since MDFS has found a path from $[w,A]$ to $[u,A]$ which does not contain $[q,B]$, there holds

$$L_{[w,A]} = [q,A] = [p,A].$$

Now, the correctness of the algorithm MDFS can easily derived from Lemma 1.2 and Lemma 1.4.

Theorem 1.2

MDFS constructs a strongly simple path from s to t, if a strongly simple path from s to t exists.

Proof:

Let

$$P = s, [v_0', B], [v_1, A], [v_1', B], \dots, [v_{r-1}', B], [v_r, A], t$$

be a strongly simple path from s to t.

It is clear that MDFS considers the edge $(s, [v_0', B])$, and performs the operation $\mathrm{PUSH}([v_0', B])$. (Note that $v_0'$ is M-free).

Hence, $[v_0', B]$, $[v_r, A]$ fulfill the assumptions of Lemma 1.2 with respect to the path

$$[v_0', B], [v_1, A], \dots, [v_{r-1}', B], [v_r, A].$$

Lemma 1.2 $\Rightarrow$

MDFS performs $\mathrm{PUSH}([v_r, A])$, and hence, $\mathrm{PUSH}(t)$.

$\Rightarrow$

MDFS constructs a path from s to t.

By Invariant 1 of Lemma 1.4, MDFS constructs only strongly simple paths. ∎

### 1.1.4 An implementation of MDFS

Now we will describe how to implement MDFS efficiently. Only two parts of the algorithm are nontrivial to implement.

1. The manipulation of $L_{[w,A]}$, $[w,A] \in V'$
2. The reconstruction of a strongly simple path P from s to t which is constructed by the algorithm.

For the solution of both subproblems it is useful not to perform the POP-operations explicitly, and to maintain the whole MDFS-tree T. This can be done as follows:

The data structure is a tree T. A pointer TOP always points to TOP(K) in T. The current MDFS-stack K is represented by the unique path from the root s of T to TOP(K) in T. For performing the operation POP, pointer TOP is changed such that it points to the unique predecessor in T. When we perform a PUSH-operation, T obtains a new leaf to which TOP points.

Invariant 2 and Invariant 3 are the key for the efficient implementation of MDFS.

Update of $L_{[w,A]}$, $[w,A] \in V'$:

Definition of $L_{[w,A]}$ $\Rightarrow$

We have only to change $L_{[w,A]}$ after a PUSH- and after a POP-operation as follows:

After PUSH($[u,A]$):

$L_{[w,A]} := \emptyset$ if $L_{[w,A]} = [u,A]$.

After POP($[u,B]$):

$L_{[w,A]} := [u,A]$ if

1. PUSH($[u,A]$) has never been performed and
2. MDFS has found a path
   $P = [w,A], Q, [u,A]$ with $[u,B] \notin Q$.

After the performance of POP($[u,B]$), eventually, MDFS has to find all nodes $[w,A]$ which fulfill Property 2 above. This can easily be done by any graph search method like depth-first search, starting in node $[u,A]$ and running the considered edges backwards. When the node $[u,B]$ is reached, a backtrack of the search is performed. But with respect to efficiency, it is useful to investigate the properties of MDFS and to refine the backward graph search.