

# 1 Shortest Augmenting Path

---

**Algorithm 1** Shortest Augmenting Path

---

**Require:** Rete  $(G, u, s, t)$

**Ensure:** flusso  $s - t$  massimo  $fMax$

$fMax \leftarrow \text{BFS}(t)$  {assegnando a ogni nodo la distanza da  $t$ , quindi  $\forall n \in V(G) \nexists t | d(n) > 0$ }

$\text{sendFlow}(fMax, s)$

**for all**  $n \in V(G)$  **do**

$n.\text{reset}()$  {elimino dati sul percorso da fare}

**end for**

**while**  $d(s) < \#V(G)$  **do**

$f \leftarrow \text{Dfs}(G, s, +\infty)$

**if**  $f \neq 0$  **then**

$\text{sendFlow}(f, t)$

**else**

**break**

**end if**

$fMax \leftarrow f + fMax$

    {valutare se qui devo inserire un reset per ogni nodo presente nel grafo}

**end while**

**return**  $fMax$

---

---

**Algorithm 2** Dfs per trovare il flusso massimo in Shortest Augmenting Path

---

**Require:** Una rete  $(G, u, s, t)$ , nodo  $start$ , valore  $f$

**Ensure:** valore del flusso inviabile {come effetto collaterale modifico le informazioni dei nodi indicando per ogni nodo la strada che deve seguire per collegare  $s$  e  $t$ }

```
if  $d(start) < \#V(G)$  then
  for all arco  $edge \in start.Edges$  do
     $n \leftarrow edge.nextNode$ 
    if  $edge.previousNode = start$  AND  $u(edge) > 0$  AND  $d(start) - 1 = d(n)$  then
       $n.setPrevious(edge)$  {aggiorno anche il previousNode di n}
       $f \leftarrow \min(f, u(e))$ 
      if  $n = t$  then
        return  $f$ 
      end if
      return Dfs( $G, n, f$ )
    end if
  end for
   $min \leftarrow +\infty$ 
  for all arco  $edge$  che entra in  $start$  do
    if flusso di  $edge$  è positivo then
       $min \leftarrow \min(min, \text{flusso di } edge)$ 
    end if
  end for
  for all arco  $edge$  che esce da  $start$  do
    if  $u(edge) > 0$  then
       $min \leftarrow \min(min, u(edge))$ 
    end if
  end for
   $d(start) \leftarrow min$ 
  if  $start = s$  then
    return Dfs( $G, start, f$ )
  else
    return Dfs( $G, start.previousNode, f$ )
  end if
end if
return 0
```

---