1 Shortest Augmenting Path

Algorithm 1 Shortest Augmenting Path

Require: Rete (G, u, s, t)
Ensure: valore del flusso massimo
  $fMax \leftarrow$BFS(t) { assegnando a ogni nodo la distanza da t}
  sendFlow$(fMax, s)$
  $esplorati \leftarrow$ lista di nodi vuota
  for all $n \in V(G)$ **do**
    $n.reset()$ {elimino dati sul percorso da fare}
  end for
  while $d(s) < \#V(G)$ **do**
    $f \leftarrow$ Dfs$(G, s, +\infty, esplorati)$
    if $f = 0$ **then**
      break
    end if
    sendFlow(f, t)
    $fMax \leftarrow f + fMax$
    for all $n \in esplorati$ **do**
      $n.reset()$
      $esplorati.remove(n)$
    end for
  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, una lista di nodi *esplorati*
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 ∈ start.Edges do
    n ← edge.nextNode
    if edge.previousNode = start ∧ u_f(edge) > 0 ∧ d(start) − 1 = d(n) then
      n.setPrevious(edge) {aggiorno anche il previousNode di n}
      f ← min(f, u_f(e))
      esplorati.add(n)
      if n = t then
        return f
      end if
      return Dfs(G, n, f, esplorati)
    end if
  end for
  min ← +∞
  for all edge ∈ δ⁺(start) do
    if u_f(edge) then
      min ← min(min, d(edge.nextNode))
    end if
  end for
  d(start) ← min + 1
  if start = s then
    return Dfs(G, start, f, esplorati)
  else
    return Dfs(G, start.previousNode, f, esplorati)
  end if
end if
return 0
```