1 Propagazione della malattia

Algorithm 1 Ricerca del massimo flusso con propagazione della malattia

Require: rete (G, u, s, t)
Ensure: valore del flusso massimo
  $noCap \leftarrow$ null
  while TRUE **do**
    $f \leftarrow$ DoBfs$(G, noCap)$
    if $f = 0$ **then**
      break
    end if
    $mom \leftarrow t$
    while $mom \neq s$ **do**
      mom.PreviousEdge.AddFlow(f)
      if $u(mom.PreviousEdge) < 0$ OR $f(mom.PreviousEdge) < 0$ **then**
        mom.Reverse(t) {da t faccio tornare come le capacità e il flusso come prima di inviarle fino a mom}
        $noCap \leftarrow mom$
        break
      end if
      if $u(mom$.PreviousEdge$= 0$ **then**
        $noCap \leftarrow mom$
      end if
      $mom \leftarrow mom$.PreviousNode
    end while
  end while
  return s.flussoUscente

Algorithm 2 DoBfs con propagazione della malattia

Require: rete (G, u, s, t), nodo *noCap*
Ensure: flusso inviabile da s al nodo t

```
coda ← coda vuota di nodi
if noCap = null then
  coda.enqueue(s)
else
  Repair(noCap) {controllo se c'è un nodo con label = noCap.label-1 e con capacità o flusso diversa da 0}
  if noCap è stato riparato AND t.PreviousNode.flussoPassante > 0 AND u(t.PreviousEdge) > 0 then
    return min(t.flussoPassante, noCap.flussoPassante)
  end if
  v ← SickPropagation(G, noCap, coda)
  if v ≠ 0 then
    return v
  end if
  for all n ∈ N(G)|n.label ≥ noCap.label do
    n.Reset()
  end for
  if coda è vuota then
    coda ← {n ∈ N(G)|n.label = (noCap.label − 1)}
  end if
  for all n ∈ coda do
    RecoverFlow(n) {Dato che è possibile che il flusso entrante sia uguale a zero dovuto a SickPropagation, torno indietro fino al primo nodo con flusso entrante positivo, quindi, ricorsivamente, inserisco il flusso entrante per tutti i nodi che ho visitato}
  end for
end if
while coda non è vuota do
  element ← coda.Dequeue()
  for all edge ← element.Edges do
    n ← edge.nextNode
    p ← edge.previousNode
    if n.visited AND p.visited (se il nodo è invalido è considerato non visitato) then
      continue
    end if
    if p = element AND u(edge) > 0 AND (n.label ≥ p.label OR¬n.valid OR noCap = null) then
      n.update(edge,p) { (label, flusso entrante, nodo precedente, nel caso riparo il nodo)}
      if n = t then
        return n.flussoPassante
      else
        coda.enqueue(n)
      end if
    else if n = element AND f(edge) > 0 AND (p.label ≥ n.label OR¬p.valid OR noCap = null) then
      p.update(edge,n)
      if p = t then
        return n.flussoPassante
      else
        coda.enqueue(p)
      end if
    end if
  end for
end while
return 0
```

Algorithm 3 SickPropagation

Require: rete (G, u, s, t), Nodo *node*, *coda*
Ensure: possibile flusso inviato verso t (partendo da n,conoscendo i valori antecedenti a n), 0 altrimenti

```
malati ← coda di nodi
malati.Enqueue(node);
while malati non è vuota do
    m ← malati.Dequeue()
    Repair(m)
    if m non è stato riparato then
        malati.enqueue(nodi con previousNode = m)
    else if m = t then
        v ← GetFlow()
        return v
    else
        coda.enqueue(m)
    end if
end while
return 0
```