

algoritmi bidirezionali

Filippo Magi

January 29, 2022

1 Ottimizzazione sono nelle ultime label

Algorithm 1 Ricerca del massimo flusso ricalcolando solo nelle ultime label

Require: Rete (G, u, s, t) .

Ensure: valore del flusso massimo

```
noCap  $\leftarrow$  null
while TRUE do
   $f \leftarrow \text{DoBfs}(G, \text{noCap})$ 
  if  $f = 0$  then
    break
  end if
   $\text{mom} \leftarrow t$ 
  while  $\text{mom} \neq s$  do
     $\text{mom.PreviousEdge.AddFlow}(t.\text{flussoPassante})$ 
    if  $u(\text{mom.PreviousEdge}) < 0$  OR  $f(\text{mom.PreviousEdge}) < 0$  then
       $\text{mom.Reverse}(t)$  {da t faccio tornare come le capacità e il flusso come
prima di inviarle fino a mom}
       $\text{noCap} \leftarrow \text{mom}$ 
      break
    end if
    if  $u(\text{mom.PreviousEdge}) = 0$  then
       $\text{noCap} \leftarrow \text{mom}$ 
    end if
     $\text{mom} \leftarrow \text{mom.previousNode}$ 
  end while
end while
return  $s.\text{FlussoUscente}$ 
```

Algorithm 2 Algoritmo DoBfs con ottimizzazione solo nelle ultime label

Require: rete (G, u, s, t) , nodo $noCap$ **Ensure:** valore del flusso inviato a t , $N(G)$ aggiornati con informazioni sul percorso da fare $coda \leftarrow$ coda vuota di nodi**if** $noCap = \text{null}$ **then** $coda.Enqueue(s)$ **else** $noCap.Valid = \text{false}$ 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 $t.PreviousNode.flussoPassante \neq 0$ AND $u(t.PreviousEdge) > 0$ **then****return** Min($t.flussoPassante$, $noCap.flussoPassante$)**end if** $coda \leftarrow n \in N(G) \parallel n.label = (noCap.label - 1)$ **for all** $n \in N(G) \parallel n.label \geq noCap.label$ **do** $n.Reset()$ **end for****end if****while** la coda non è vuota **do** $element \leftarrow coda.dequeue()$ **if** $element.valid$ **then** {label valida}**for all** $edge \in element.Edges$ **do** $n \leftarrow edge.nextNode$ $p \leftarrow edge.previousNode$ **if** $p = element$ AND $u(edge) > 0$ AND $(\neg n.visited \text{ OR } \neg n.valid)$ **then** $n.update(p, edge)$ {label, flusso entrante e nodo precedente, nel caso sia necessario "riparo" il nodo} $edge.Reversed = \text{false}$ {per capire se devo inviare o ricevere flusso}**if** $n = t$ **then****return** $n.flussoPassante$ **else** $coda.Enqueue(n)$ **end if****else if** $n = element$ AND $f(edge) > 0$ AND $(\neg p.visited \text{ OR } \neg p.valid)$ **then** $p.update(n, edge)$ $edge.reversed = \text{true}$ **if** $p = t$ **then****return** $p.flussoPassante$ **else** $coda.Enqueue(p)$ **end if****end if****end for****end if****end while****return** 0
