algoritmi bidirezionali

Filippo Magi

February 16, 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
  $malati \leftarrow$ pila di nodi vuota
  while TRUE **do**
    $f \leftarrow$ DoBfs$(G, noCap)$
    if $f = 0$ **then**
      break
    end if
    $mom \leftarrow t$
    while $mom \neq s$ **do**
      $mom.PreviousEdge.AddFlow(t.flussoPassante)$
      if $u(mom.PreviousEdge) = 0$ **then**
        $malati$.push(mom)
      end if
      $mom \leftarrow mom$.previousNode
    end while
  end while
  return s.FlussoUscente

Algorithm 2 Algoritmo DoBfs con ottimizzazione solo nelle ultime label

Require: rete (G, u, s, t), pila di nodi *malati*
Ensure: valore del flusso inviato a t, $N(G)$ aggiornati con informazioni sul percorso da fare

```
coda ← coda vuota di nodi
if malti.isEmpty then
  coda.Enqueue(s)
  for all n ∈ V(G) do
    n.Reset()
  end for
else
  repaired ← true
  p ← null
  while ¬malati.isEmpty do
    n ← malati.pop()
    UpdateInFlow(n, p) {se p ≠ null, aggiorno i nodi tra p ed n con il nuovo valore di InFlow, passando da 0 a un valore positivo}
    Repair(n){controllo se c'è un nodo con label = noCap.label-1 e con capacità o flusso diversa da 0}
    if non sono riuscito a riparare n then
      repaired ←false
      break
    end if
    p ← n
  end while
  if repaired then
    f ← UpdateInFlow (t, n)
    return f
  end if
  coda ← x ∈ V(G)‖x.label = (n.label − 1)
  for all x ∈ V(G)‖x.label ≥ n.label do
    x.Reset()
  end for
end if
```

```
while la coda non è vuota do
  element ← coda.dequeue()
  if element.valid then {label valida}
    for all edge ∈ element.Edges do
      n ← edge.nextNode
      p ← edge.previousNode
      if  p = element ∧ u_t(edge) > 0 ∧ (¬n.visited ∨ ¬n.valid) then {per
capire se il nodo è stato visitato o meno, controllo che inflow > 0}
        n.update(p,edge) {label, flusso entrante e nodo precedente, nel caso
sia necessario "riparo" il nodo}
        edge.Reversed = 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 ∧ f(edge) > 0 ∧ (¬p.visited ∨ ¬p.valid) then
        p.update(n,edge)
        edge.reversed = true
        if p = t then
          return p.flussoPassante
        else
          coda.Enqueue(p)
        end if
      end if
    end for
  end if
end while
return 0
```