SickPropagation

Filippo Magi

January 3, 2022

1 Propagazione della malattia

Algorithm 1 Ricerca del massimo flusso a costo minimo con propagazione della malattia

Require: grafo dei residui $\overleftrightarrow{G} = \{V(G), E(G) \cup \{\overleftarrow{e} : e \in E(G)\}$.
Ensure: valore del flusso massimo

```
s ← SourceNode di G↔
t ← SinkNode di G↔
noCap ← null
loop
  f ← DoBfs(grafo,noCap)
  if f = 0 then
    break
  end if
  mom ← t
  while mom ≠ s do
    aggiorno capacità o flusso del nodo m a seconda del suo predecessore
    if capacità dell'arco (m,predecossore di m) = 0 then
      noCap ← mom
    end if
    mom ← predecessore di mom
  end while
end loop
return flusso uscente da s
```

Algorithm 2 DoBfs con propagazione della malattia

Require: grafo dei residui $\overleftrightarrow{G}$, nodo noCap
Ensure: flusso inviato al nodo t, $\overleftrightarrow{G}$ aggiornato

```
coda ← coda vuota di nodi
if noCap = null then
  coda.enqueue(sourceNode di G↔)
else
  t ← SinkNode di G↔
  provo a riparare noCap {controllo se c'è un nodo con label = noCap.label-1
  e con capacità o flusso diversa da 0, aggiornando i dati}
  if noCap è stato riparato then
    f ← flusso massimo consentito nel percorso da t a s
    if f ≠ 0 then
      return f
    end if
  end if
  v ← SickPropagation(grafo,noCap,coda)
  if v ≠ 0 then
    return v
  end if
  confermo di non aver esplorato i nodi con label ≤ noCap.label
  if coda è vuota then
    coda ← nodi di G↔ con label = (noCap.label - 1)
  end if
end if
while la coda non è vuota do
  element ← code.Dequeue()
  for all edge ← archi che escono ed entrano in element do
    n ← nodo che entra da edge
    p ← nodo che esce da edge
    if n ed p sono stati visitati (se il nodo è invalido è considerato non
    visitato) then
      continue
    end if
    if p = element AND capacità di edge > 0 AND (n è successivo a p OR
    n non è valido OR noCap = null) then
      aggiorno i dati di n (label,visitato, nodo precedente, nel caso riparo il
      nodo)
      if n è SinkNode di G↔ then
        return flusso massimo da s a t, secondo il percorso scelto (dai vari
        previousNode all'interno dei nodi)
      else
        coda.enqueue(n)
      end if
    else if n = element AND flusso di edge > 0 AND p non è stato visitato
    then
      aggiorno i dati di p
      if p è SinkNode di G↔ then
        return flusso massimo da s a t, secondo il percorso scelto
      else
        coda.enqueue(p)
      end if
    end if
  end for
end while
return 0
```

Algorithm 3 SickPropagation

Require: grafo dei residui $\overleftrightarrow{G}$, Nodo node, coda di nodi
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()
    provo a riparare m
    if il nodo non è stato riparato then
        malati.enqueue(nodo con previousNode = m)
    else if m è SinkNode di G↔ then
        v ← ricorsivamente, dal t vado a recupeare il flusso massimo inviabile
        secondo il percorso deciso
        return v
    else
        coda.enqueue(m)
    end if
end while
return 0
```