

SickPropagation

Filippo Magi

December 29, 2021

1 Propagazione della malattia

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

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

$s \leftarrow \text{SourceNode di } \vec{G}$
 $t \leftarrow \text{SinkNode di } \vec{G}$
 $\text{noCap} \leftarrow \text{null}$

loop
 $f \leftarrow \text{DoBfs}(\text{grafo}, \text{noCap})$
 if $f = 0$ **then**
 break
 end if
 $\text{mom} \leftarrow t$
 while $\text{mom} \neq s$ **do**
 aggiorno capacità o flusso del nodo m a seconda del suo predecessore
 if capacità dell'arco $(m, \text{predecessore di } m) = 0$ **then**
 $\text{noCap} \leftarrow \text{mom}$
 end if
 $\text{mom} \leftarrow \text{predecessore di } \text{mom}$
 end while
end loop
return flusso uscente da s

Algorithm 2 DoBfs con propagazione della malattia

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

```
coda  $\leftarrow$  coda vuota di nodi
if noCap = null then
    coda.enqueue(sourceNode di  $\vec{G}$ )
else
    t  $\leftarrow$  SinkNode di  $\vec{G}$ 
    provo a riparare 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 il flusso entrante nel predecessore di t  $\neq$  0
    AND capacità dell'arco (t, predecessore di t)  $>$  0 then
        return Min(flusso entrante di t, flusso passante per noCap)
    end if
    v  $\leftarrow$  SickPropagation(grafo,noCap,coda)
    if v  $\neq$  0 then
        return v
    end if
    cancello il flusso entrante nei nodi con label  $\leq$  noCap.label
    if coda è vuota then
        coda  $\leftarrow$  nodi di  $\vec{G}$  con label = (noCap.label - 1)
    end if
    for all nodo n  $\in$  coda do
        recupera flusso entrante dall'ultimo nodo che lo ha {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, ricorsi-
        vamente, inserisco il flusso entrante per tutti i nodi che ho visitato}
    end for
end if
while la coda non è vuota do
    element  $\leftarrow$  coda.Dequeue()
    for all edge  $\leftarrow$  archi che escono ed entrano in element do
        n  $\leftarrow$  nodo che entra da edge
        p  $\leftarrow$  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
            aggiorni i dati di n (label, flusso entrante, nodo precedente, nel caso
            riparo il nodo)
            if n è SinkNode di  $\vec{G}$  then
                return flusso entrante in n
            else
                coda.enqueue(n)
            end if
        else if n = element AND flusso di edge  $>$  0 AND p non è stato visitato
        then
            aggiorni i dati di p
            if p è SinkNode di  $\vec{G}$  then
                return flusso entrante in n
            else
                coda.enqueue(p)
            end if
        end if
    end for
end while
```

Algorithm 3 SickPropagation

Require: grafo dei residui \vec{G} , Nodo node, coda di nodi
Ensure: possibile flusso inviato verso t (partendo da n, conoscendo i valori antecedenti a n), 0 altrimenti
malati \leftarrow coda di nodi
malati.Enqueue(node);
while malati non è vuota **do**
 m \leftarrow 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 \vec{G} **then**
 v \leftarrow ricorsivamente, dal sourceNode di \vec{G} ottengo il flusso massimo che posso inviare, salvando il flusso entrante per ogni arco (recuperato tramite previousNode) che attraverso
 return v
 else
 coda.enqueue(m)
 end if
end while
return 0
