algoritmi bidirezionali

Filippo Magi

January 21, 2022

1 ShortestAugmentingPath

1.1 FlowFordFulkerson

1.2 Dfs

Algorithm 1 Ricerca del flusso massimo

Require: grafo dei residui $\overleftrightarrow{G} = \{V(G), E(G) \cup \{\overleftarrow{e} : e \in E(G)\}$, nodo Source s, nodo destinazione t .

Ensure: valore del flusso massimo di $\overleftrightarrow{G}, E(\overleftrightarrow{G})$ aggiornato

1: $startSource \leftarrow s$
2: $startSink \leftarrow t$
3: $fMax \leftarrow$ Bfs(s){faccio partire da s una bfs, cercando un percorso e sopratutto indicando la distanza che c'è fra un certo nodo e source}
4: sendFlow($t, fMax$) {invio il flusso dal percorso indicato tramite previousNode da t verso s con il valore fMax,nel mentre che procedo cancello le informazioni nei nodi esplorati (tranne la distanza)}
5: $f \leftarrow$ Bfs(t) {bfs da t verso s, trovo un percorso salvato da NextNode e sopratutto trovo la distanza tra un certo nodo e t}
6: sendFlow(s, f)
7: $fMax \leftarrow f + fMax$
8: **for all** var n in $N(\overleftrightarrow{G})$ **do**
9: n.Reset() {cancello indicazioni su un possibile percorso da fare}
10: **end for**
11: $fso \leftarrow +\infty, fsi \leftarrow +\infty$
12: **while** $f \neq 0$ AND distanza tra s e $t >$ nodi di $\overleftrightarrow{G}$, AND distanza tra t ed $s >$ nodi di $\overleftrightarrow{G}$ **do** $(fso, fsi, startSource, startSink) \leftarrow$ Dfs $(\overleftrightarrow{G}, startSource, startSink, fso, fsi, s, t)$
13: **if** $startSink = startSource$ AND $startSink \neq null$ **then**
14: $f \leftarrow \min(fso, fsi)$
15: sendFlow($startSink, f$)
16: $fso \leftarrow +\infty, fsi \leftarrow +\infty$
17: $startSource \leftarrow s, startSink \leftarrow t$
18: **else if** $startSink = s$ **then**
19: $f \leftarrow fsi$
20: sendFlow($startSink, f$)
21: $fsi \leftarrow +\infty$
22: $startSink \leftarrow t$
23: **else if** $startSource = t$ **then**
24: $f \leftarrow fso$
25: sendFlow($startSource, f$)
26: $fso \leftarrow +\infty$
27: $startsource \leftarrow s$
28: **else**
29: **break**
30: **end if**
31: $fMax \leftarrow f + fMax$
32: {valutare se serve fare un reset per ogni nodo presente nel grafo}
33: **end while**
34: **return** fMax

Algorithm 2 Dfs

Require: grafo dei residui $\overleftrightarrow{G}$, *startSource* e *startSink* rispettivamente il nodo di partenza della parte di Source e di Sink, *sourceFlow* e *sinkFlow* rispettivamente il valore del flusso massimo inviabile dalla parte di Source e di Sink, nodo sorgente s, nodo destinazione t

Ensure: (massimo valore di flusso inviabile da parte di source, massimo valore di flusso inviabile da parte di Sink, Nodo di arrivo dalla parte di Source, nodo di arrivo dalla parte di Sink)

1: **if** $startSource = startSink$ **then**
2: **return** $(sourceFlow, sinkFlow, startSource, startSink)$
3: **end if**
4: **if** distanza tra $startSink$ e $s <$ numero di nodi di $N(\overleftrightarrow{G})$ AND distanza tra $startSource$ e $t <$ numero di nodi di $N(\overleftrightarrow{G})$ **then**
5: **for all** arco $edge$ in $startSource.Edges$ **do**
6: $n \leftarrow edge.NextNode$
7: $p \leftarrow edge.PreviousNode$
8: **if** $startSource = p$ AND (distanza tra n e t) = ((distanza tra p e t) - 1) AND capacità di $edge > 0$ **then**
9: $sourceFlow \leftarrow \min(sourceFlow$, capacità di $edge)$
10: arco precedente di $n \leftarrow edge$ {salvo anche il nodo precedente}
11: **if** n è già stato precedentemente esplorato dalla parte di Sink **then**
12: **return** $(sourceFlow, sinkFlow, n, n)$
13: **end if**
14: **if** $n = t$ **then**
15: **return** $(sourceFlow, sinkFlow, n, startSink)$
16: **end if**
17: **return** SinkDfs$(\overleftrightarrow{G}, n, startSink, sourceFlow, sinkFlow, s, t)$
18: **end if**
19: **end for**
20: $minDistance \leftarrow +\infty$
21: **for all** arco $edge \in startSource.Edges$ **do**
22: **if** $edge.PreviousNode = startSource$ AND capacità di $edge > 0$ **then**
23: $minDistance \leftarrow \min(minDistance$, distanza tra nodo successivo di $edge$ e t
24: **end if**
25: **end for**
26: distanza tra $startSource$ e $t \leftarrow minDistance + 1$
27: **if** $startSource = s$ **then**
28: $mom \leftarrow startsource$
29: **else**
30: $mom \leftarrow$ nodo precedente di $startSource$
31: **end if**
32: $startSource.Reset()$
33: **return** Dfs$(\overleftrightarrow{G}, mom, startSink, sourceFlow, sinkFlow, s, t)$
34: **end if**
35: **return** $(0, 0, null, null)$

Algorithm 3 SinkDfs

Require: grafo dei residui $\overleftrightarrow{G}$, *startSource* e *startSink* rispettivamente il nodo di partenza della parte di Source e di Sink, *sourceFlow* e *sinkFlow* rispettivamente il valore del flusso massimo inviabile dalla parte di Source e di Sink, nodo sorgente s, nodo destinazione t

Ensure: (massimo valore di flusso inviabile da parte di source, massimo valore di flusso inviabile da parte di Sink, Nodo di arrivo dalla parte di Source, nodo di arrivo dalla parte di Sink)

if $startSource = startSink$ **then**
  return $(sourceFlow, sinkFlow, startSource, startSink)$
end if
if distanza tra $startSink$ e $s <$ numero di nodi di $N(\overleftrightarrow{G})$ AND distanza tra $startSource$ e $t <$ numero di nodi di $N(\overleftrightarrow{G})$ **then**
  for all arco $edge$ in $startSink.Edges$ **do**
    $n \leftarrow edge.NextNode$
    $p \leftarrow edge.PreviousNode$
    if $startSink = n$ AND (distanza tra p e t) = ((distanza tra n e t) - 1) AND capacità di $edge > 0$ **then**
      $sourceFlow \leftarrow \min(sinkFlow$, capacità di $edge)$
      arco precedente di $p \leftarrow edge$ {salvo anche il nodo precedente}
      if p è già stato precedentemente esplorato dalla parte di Sink **then**
        return $(sourceFlow, sinkFlow, p, p)$
      end if
      if $p = n$ **then**
        return $(sourceFlow, sinkFlow, startSource, p)$
      end if
      return Dfs$(\overleftrightarrow{G}, startSource, p, sourceFlow, sinkFlow, s, t)$
    end if
  end for
  $minDistance \leftarrow +\infty$
  for all arco $edge \in startSink.Edges$ **do**
    if $edge.NextNode = startSink$ AND capacità di $edge > 0$ **then**
      $minDistance \leftarrow \min(minDistance$, distanza tra nodo precedente di $edge$ e t
    end if
  end for
  distanza tra $startSink$ e $t \leftarrow minDistance + 1$
  if $startSink = t$ **then**
    $mom \leftarrow startSink$
  else
    $mom \leftarrow$ nodo successore di $startSink$
  end if
  $startSink.Reset()$
  return SinkDfs$(\overleftrightarrow{G}, startSource, mom, sourceFlow, sinkFlow, s, t)$
end if
return $(0, 0, null, null)$