CAMMINI MINIMI DA SINGOLA SORGENTE IN GRAFI ORIENTATI

LABEL SETTING VS LABEL CORRECTING

- SIA DATO UN GRAFO $G = (V, E)$ CON FUNZIONE PESO $w: E \rightarrow \mathbb{R}$ E SORGENTE $s \in V$.

- LABEL SETTING E LABEL CORRECTING SONO DUE TECNICHE PER LA RISOLUZIONE DI PROBLEMI DI CAMMINI MINIMI

- IN ENTRAMBI I CASI VIENE MANTENUTA UNA STIMA $d: V \longrightarrow \mathbb{R}$ DELLE DISTANZE DALLA SORGENTE s TALE CHE

  - $d[v] \geq \delta(s, v)$
  - d DECRESCE MONOTONICAMENTE SU OGNI $v \in V$

- NEL CASO DEGLI ALGORITMI BASATI SULLA TECNICA LABEL CORRECTING (ES, BELLMAN-FORD), I VALORI CORRETTI DELLE DISTANZE DIVENTANO NOTI SOLA ALLA FINE DELL' ESECUZIONE

- NEL CASO DEGLI ALGORITMI BASATI SULLA TECNICA LABEL SETTING (ES, DIJKSTRA, ALGORITMO PER GRAFI ACICLICI), I VALORI CORRETTI DELLE DISTANZE DIVENTANO NOTI GIÀ NEL CORSO DELL' ESECUZIONE

ALGORITMI BASATI SULLA TECNICA LABEL SETTING

GENERIC-ALGORITHM (G, s, w)

INITIALIZE-SINGLE-SOURCE(G, s)

$S := \emptyset$ // rappresenta l'insieme dei nodi x
// per i quali e' noto che $d[x] = \delta_G(s, x)$

while $\exists\, v \in V[G] \setminus S$ tale che $d[v] = \delta_G(s, v)$ do

- sia $v \in V[G] \setminus S$ tale che $d[v] = \delta_G(s, v)$

SCAN$(v; G, w)$

$S := S \cup \{v\}$

- I VALORI $d[v]$ SONO CONSOLIDATI NEL MOMENTO IN CUI I NODI v ENTRANO A FAR PARTE DI S

procedure INITIALIZE-SINGLE-SOURCE(G, s)

for $v \in V[G]$ **do**

$d[v] := +\infty$

$Pred[v] := NIL$

$d[s] := 0$

procedure SCAN $(u; G, w)$

  for $v \in Adj_G[u]$ **do**

    RELAX $(u, v; w)$

procedure RELAX $(u, v; w)$

  if $d[v] > d[u] + w(u,v)$ **then**

    $d[v] := d[u] + w(u,v)$

    $Pred[v] := u$

PROPRIETÀ

NEL CORSO DELL'ESECUZIONE DEL GENERIC-ALGORITHM d DECRESCE MONOTONICAMENTE SU OGNI $v \in V$

DIMOSTRAZIONE

E' SUFFICIENTE OSSERVARE CHE IL VALORE DI $d[v]$ PUÒ CAMBIARE SOLO A SEGUITO DELL'ESECUZIONE DI RELAX $(u, v; w)$, PER QUALCHE $u \in V$, E PUÒ SOLTANTO DIMINUIRE. $\square$

PROPRIETÀ DEL LIMITE SUPERIORE

NEL CORSO DELL'ESECUZIONE DEL GENERIC-ALGORITHM,

$d[v] \geq \delta(s,v)$, PER OGNI $v \in V$. (*)

DIMOSTRAZIONE

PER INDUZIONE SUL NUMERO DI CHIAMATE A RELAX

CASO BASE: SUBITO DOPO L'INIZIALIZZAZIONE,

(*) VALE BANALMENTE

PASSO INDUTTIVO (DOPO LA CHIAMATA RELAX$(u,v;w)$)

- SE $d[v]$ RIMANE INVARIATA, ALLORA (*) È VERA PER L'IPOTESI INDUTTIVA
- ALTRIMENTI:

$$d[v] = d[u] + w(u,v) \underset{\text{IPOTESI INDUTTIVA}}{\geq} \delta(s,u) + w(u,v) \geq \delta(s,v), \quad \square$$

- GLI ALGORITMI FONDATI SULLA TECNICA LABEL SETTING BASANO LA LORO CORRETTEZZA SULLA "PROPRIETÀ DELLA CONVERGENZA":

SE

- $\pi: s \rightsquigarrow u \to v$ E' UN CAMMINO MINIMO IN G
- PRIMA DI UNA CERTA CHIAMATA RELAX$(u, v; w)$ VALE $d[u] = \delta(s,u)$

ALLORA

DOPO TALE CHIAMATA RELAX$(u, v; w)$ VALE ANCHE $d[v] = \delta(s,v)$

DIMOSTRAZIONE DELLA PROPRIETÀ DELLA CONVERGENZA

DOPO L'ESECUZIONE DI RELAX $(u, v; w)$ SI HA:

PROPRIETÀ DEL LIMITE SUPERIORE

$$\delta_G(s,v) \leq d[v] \leq d[u] + w(u,v)$$
$$= \delta_G(s,u) + w(u,v)$$
$$= w(\pi)$$
$$= \delta_G(s,v)$$

PERTANTO: $d[v] = \delta_G(s,v)$ $\square$

<u>procedure</u> RELAX $(u, v; w)$

    <u>if</u> $d[v] > d[u] + w(u,v)$ <u>then</u>

        $d[v] := d[u] + w(u,v)$

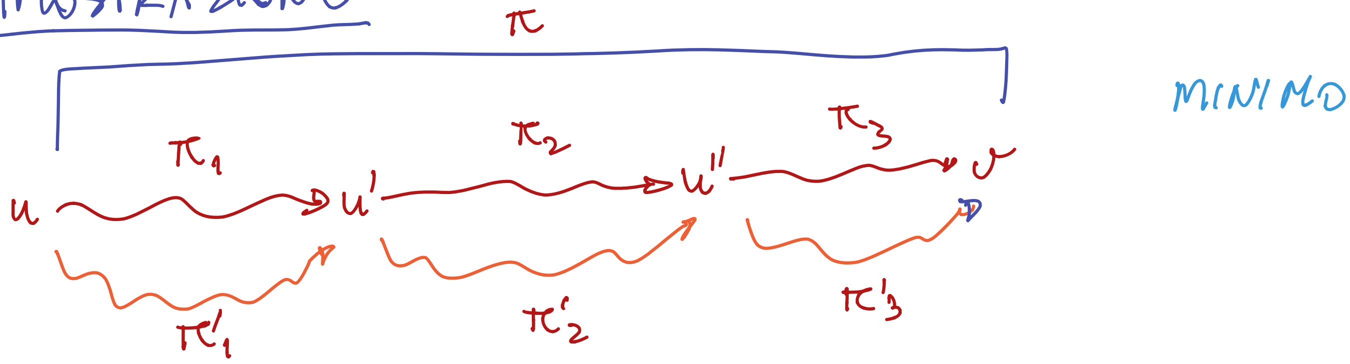
        $Pred[v] := u$

PROPRIETÀ DELLA SOTTOSTRUTTURA OTTIMA DEI CAMMINI MINIMI

SIA $\pi = \pi_1 ; \pi_2 ; \pi_3$ UN CAMMINO MINIMO IN (G, w), CON $w : E \to \mathbb{R}$.

ALLORA I SOTTOCAMMINI π_1, π_2 E π_3 SONO MINIMI.

DIMOSTRAZIONE



- SE ESISTESSERO $\pi'_1 : u \rightsquigarrow u'$

$\pi'_2 : u' \rightsquigarrow u''$

$\pi'_3 : u'' \rightsquigarrow v$

TALI CHE

$$w(\pi'_i) \leq w(\pi_i) \qquad (\forall i = 1, 2, 3)$$

E

$$w(\pi'_{j_0}) < w(\pi_{j_0}) \qquad (\text{PER QUALCHE } j_0)$$

ALLORA, POSTO $\pi' = \pi'_1 ; \pi'_2 ; \pi'_3$, SI AVREBBE:

$$\begin{aligned} w(\pi') &= w(\pi'_1) + w(\pi'_2) + w(\pi'_3) \\ &< w(\pi_1) + w(\pi_2) + w(\pi_3) \\ &= w(\pi) \end{aligned}$$

CONTRADDICENDO LA MINIMALITA' DI π. $\square$

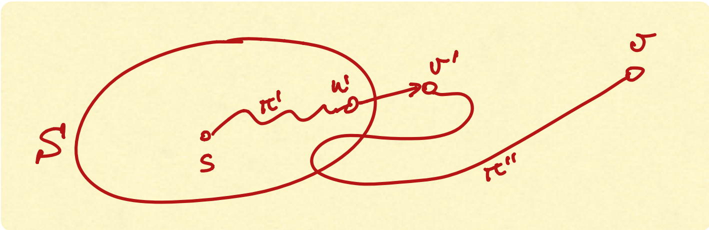
PROPRIETÀ

IL GENERIC-ALGORITHM TERMINA CORRETTAMENTE (SE IL GRAFO AMMETTE CAMMINI MINIMI DALLA SORGENTE).

DIMOSTRAZIONE

SE $S \neq V$, SIA $v \in V \setminus S$.

- SE v NON E' RAGGIUNGIBILE DA s, ALLORA $d[v] = +\infty = \delta_G(s,v)$.



- ALTRIMENTI, SIA

$$\pi : \pi' ; (u', v') ; \pi''$$

UN CAMMINO MINIMO DA s A v TALE CHE

- TUTTI I NODI DI π' SONO IN S
- $v' \notin S$

- PER LA PROPRIETA' DELLA SOTTOSTRUTTURA OTTIMA DEI CAMMINI MINIMI, IL CAMMINO $\pi'; (u', v')$ E' MINIMO.

PERTANTO PER LA PROPRIETA' DI CONVERGENZA SI HA:

$$d[v'] = \delta(s, v')$$

NE CONSEGUE CHE IL GENERIC-ALGORITHM NON PUÒ CHE TERMINARE CORRETTAMENTE. $\square$

GENERIC-ALGORITHM (G, s, w)

INITIALIZE-SINGLE-SOURCE (G, s)

$S := \emptyset$ // rappresenta l'insieme dei nodi x
// per i quali e' noto che $d[x] = \delta_G(s, x)$

while $\exists\, v \in V[G] \setminus S$ tale che $d[v] = \delta_G(s, v)$ **do**

- sia $v \in V[G] \setminus S$ tale che $d[v] = \delta_G(s, v)$

SCAN $(v; G, w)$

$S := S \cup \{v\}$

$\Uparrow$

COME SELEZIONARE v
IN MANIERA EFFICIENTE?

NEI DUE CASI IN CUI

- G E' ACICLICO

OPPURE

- $w : E \to \mathbb{R}^+_0$

CIO' E' POSSIBILE.

CAMMINI MINIMI DA SINGOLA SORGENTE IN GRAFI ACICLICI

procedure DAG-SHORTEST-PATHS'(G, s, w)

INITIALIZE-SINGLE-SOURCE(G, s)

$S := \emptyset$

while $V[G] \setminus S \neq \emptyset$ **do**

- sia $v \in V[G] \setminus S$ un nodo i cui predecessori stiano in S

SCAN$(v; G, w)$

$S := S \cup \{v\}$

COME SELEZIONARE v IN MANIERA EFFICIENTE?

procedure DAG-SHORTEST-PATHS (G, s, w)

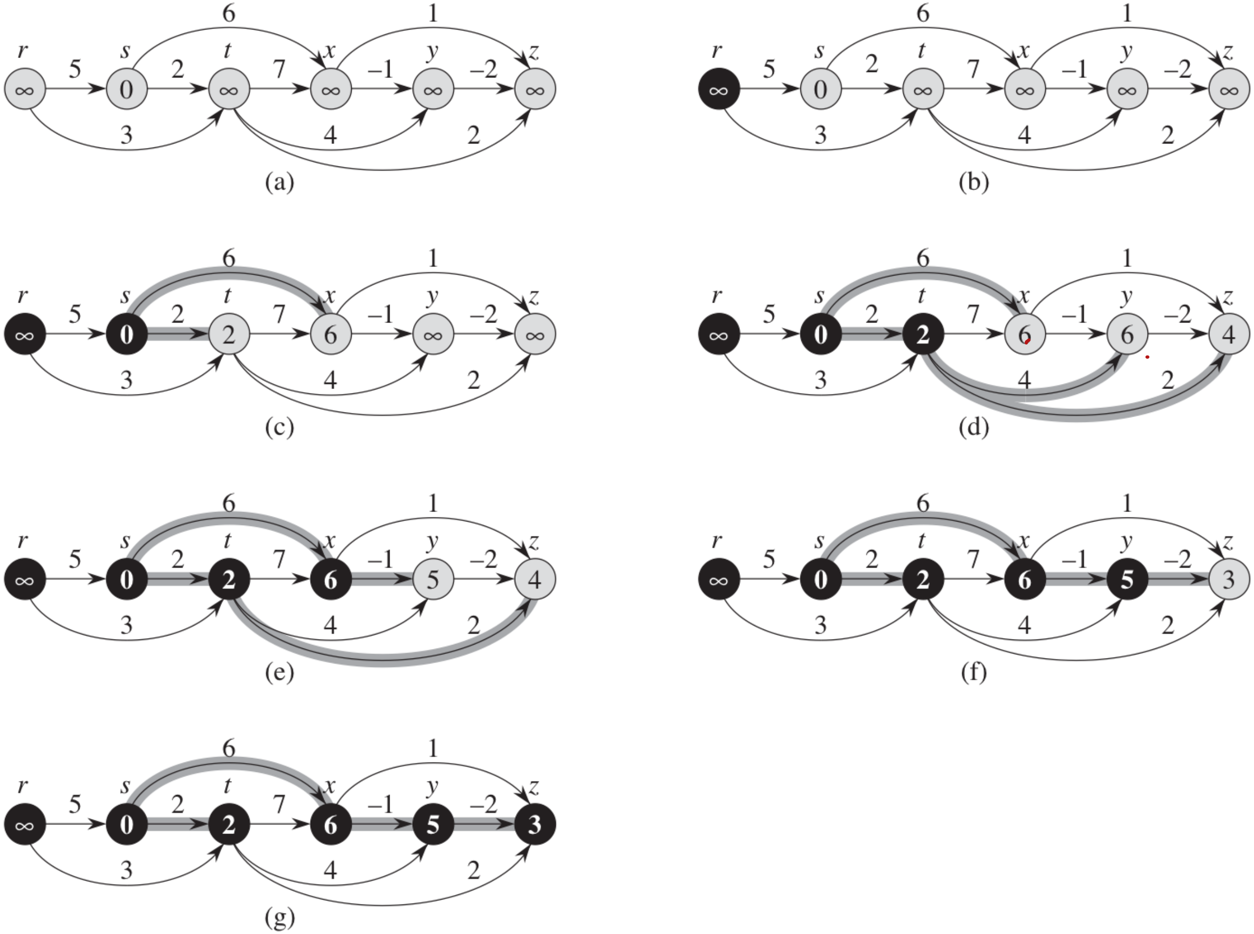
INITIALIZE-SINGLE-SOURCE (G, s)

- sia $\prec$ un ordinamento topologico di $G = (V, E)$

for $v \in V[G]$ seguendo l'ordinamento $\prec$ **do**

SCAN $(v; G, w)$

ESEMPIO



UN' IMPLEMENTAZIONE ALTERNATIVA

Preprocessing (G)

$V := \mathcal{V}[G]$; $E := \mathcal{E}[G]$;

for $i := 1$ **to** $|V|$ **do**

predecessors$[i] := 0$

let L be an empty list

for $(i,j) \in E$ **do**

predecessors$[j] :=$ predecessors$[j] + 1$

for $i := 1$ **to** $|V|$ **do**

if predecessors$[i] = 0$ **then**

add i to L ;

return predecessors, L ;

```
procedure DAG-SHORTEST-PATHS'(G, 1, w)
  [predecessors, L] := Preprocessing(G)
  INITIALIZE-SINGLE-SOURCE(G, 1)
  while L ≠ ∅ do
      i := Extract-head[L]
      for j ∈ Adj_G[i] do
          RELAX(i, j, w)
          predecessors[j] := predecessors[j] − 1
          if predecessors[j] = 0 then
              Insert(j, L)
```

COMPLESSITA'

procedure DAG-SHORTEST-PATHS (G, s, w)

INITIALIZE-SINGLE-SOURCE(G, s) — $O(V)$

– sia $\prec$ un ordinamento topologico di S — $O(V+E)$

for $v \in V[G]$ seguendo l'ordinamento $\prec$ **do** — $O(V+E)$

SCAN $(v; G, w)$

$O(V+E)$

ALGORITMO DI DIJKSTRA

- SUPPONIAMO CHE $w: E \to \mathbb{R}_0^+$ (CIOÈ CHE NON CI SIANO ARCHI NEGATIVI)

GENERIC-ALGORITHM (G, s, w)

INITIALIZE-SINGLE-SOURCE(G, s)

$S := \emptyset$

while $\exists\, v \in V[G] \setminus S$ tale che $d[v] = \delta_G(s, v)$ do

- sia $v \in V[G] \setminus S$ tale che $d[v] = \delta_G(s, v)$

SCAN$(v; G, w)$

$S := S \cup \{v\}$

$\Uparrow$

COME SELEZIONARE v IN MANIERA EFFICIENTE?

PROPRIETÀ GREEDY

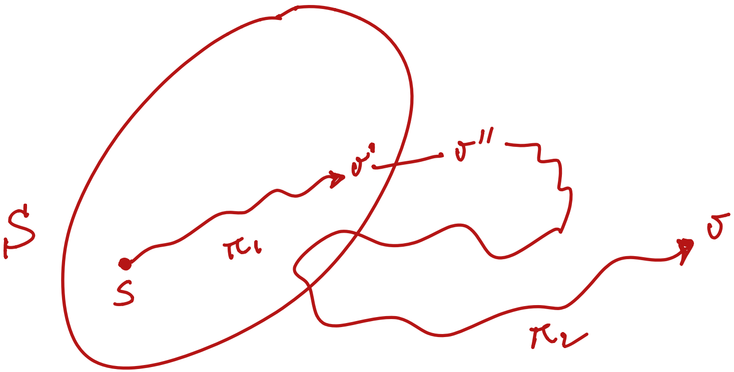
SE $v \in V \setminus S$ È TALE CHE $d[v] = \min\{d[u] \mid u \in V \setminus S\}$, ALLORA $d[v] = \delta_G(s,v)$.

DIMOSTRAZIONE

PER ASSURDO.

- SIA $v \in V \setminus S$ TALE CHE
  - $d[u] = \delta_G(s,u)$, PER OGNI $u \in S$
  - $d[v] = \min\{d[u] \mid u \in V \setminus S\}$
  - $d[v] > \delta_G(s,v)$ (PER ASSURDO)

- SIA $\pi : \underbrace{s \overset{\pi_1}{\rightsquigarrow} v'}_{\text{IN } S} \longrightarrow \underbrace{v''}_{\in V\setminus S} \overset{\pi_2}{\rightsquigarrow} v$ (MINIMO)



- TUTTI I NODI DI π_1 SONO IN S (DUNQUE $v' \in S$)
- $v'' \notin S$
- $d[v] = \min\{d[u] \mid u \in V\setminus S\}$
- $d[v] > \delta_G(s,v) = w(\pi)$

- DUNQUE

$$d[v] > \delta_G(s,v) = w(\pi)$$

$$= \underbrace{w(\pi_1) + w(v', v'')}_{} + w(\pi_2)$$

$$= \delta(s, v'') + w(\pi_2)$$

(PER LA PROPRIETÀ DELLA SOTTOSTRUTTURA OTTIMA)

$$\geq \delta(s, v'')$$ (POICHÉ $w(\pi_2) \geq 0$)

$$= d[v''],$$

(PER LA PROPRIETÀ DELLA CONVERGENZA)

CONTRADDICENDO LA MINIMALITÀ DI $d[v]$. $\square$

procedure DIJKSTRA 1 (G, s, w)

INITIALIZE-SINGLE-SOURCE(G, s)

$S := \emptyset$

while $V[G] \setminus S \neq \emptyset$ **do**

- sia $v \in V[G] \setminus S$ tale che $d[v] = \min\{d[u] \mid u \in V \setminus S\}$

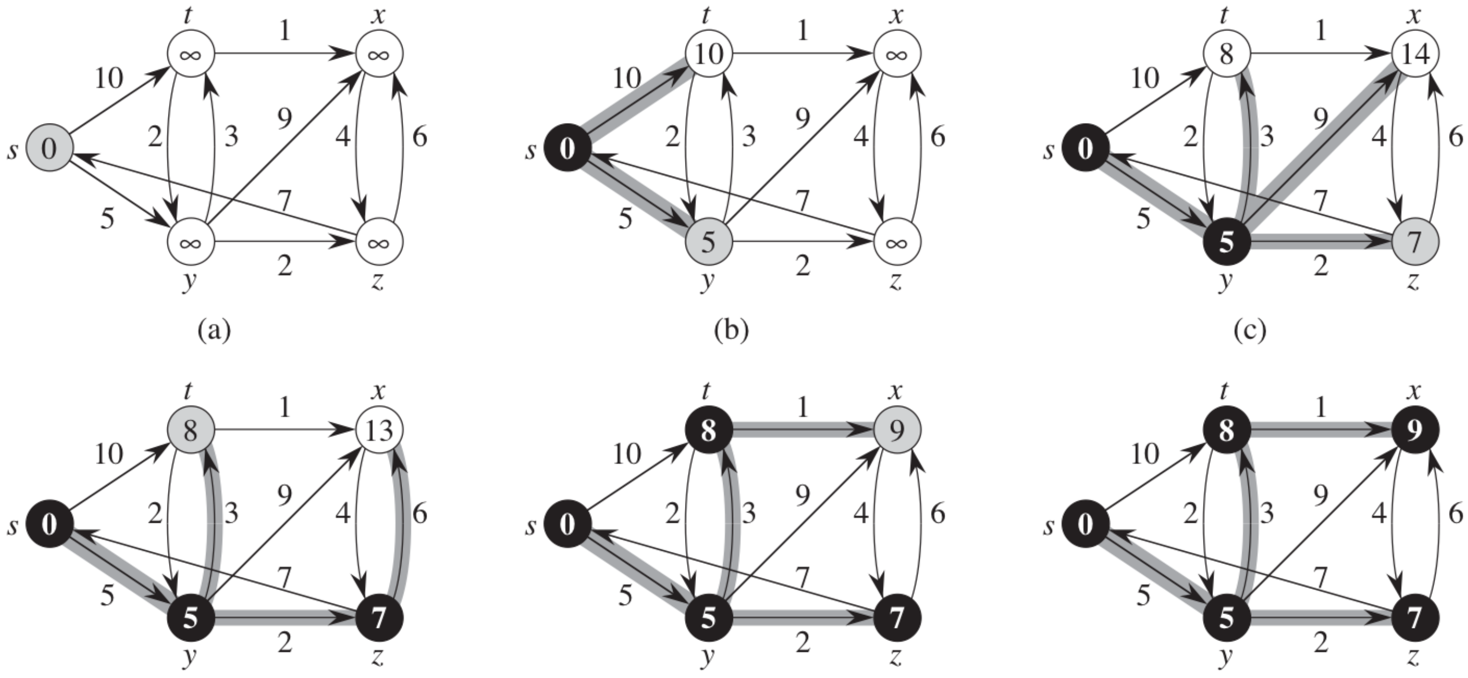
SCAN$(v; G, w)$

$S := S \cup \{v\}$

$\Uparrow$

COME SELEZIONARE v
IN MANIERA EFFICIENTE?

ESEMPIO



(a) (b) (c)

(d) (e) (f)

```
procedure DIJKSTRA (G, s, w)
INITIALIZE-SINGLE-SOURCE (G, s)
Q := BUILD-HEAP (V[G], d)
while Q ≠ ∅ do
    v := EXTRACT-MIN (Q, d)
    SCAN (v; G, w)
```

COMPLESSITA'

procedure DIJKSTRA (G, s, w)

INITIALIZE-SINGLE-SOURCE (G, s) — $O(V)$

$Q :=$ BUILD-HEAP $(V[G], d)$ — $O(V)$

while $Q \neq \emptyset$ **do**

  $v :=$ EXTRACT-MIN (Q, d)

  SCAN $(v; G, w)$

$|V| \cdot$ costo (EXTRACT-MIN)

$+ \; |E| \cdot$ costo (DECREASE-KEY)

COMPLESSITA' CON VARIE IMPLEMENTAZIONI DELL'HEAP

procedure DIJKSTRA (G, s, w)

INITIALIZE-SINGLE-SOURCE (G, s)

$Q :=$ BUILD-HEAP $(V[G], d)$

while $Q \neq \emptyset$ **do**

$v :=$ EXTRACT-MIN (Q, d)

SCAN $(v; G, w)$

ARRAY	HEAP BINARIO	HEAP DI FIBONACCI
$O(V)$	$O(V)$	$O(V)$
$O(1)$	$O(V)$	$O(V)$
$O(V^2)$	$O(V \log V)$	$O(V \log V)$
$O(E)$	$O(E \log V)$	$O(E)$
$O(V^2)$	$O((V+E) \log V)$	$O(E + V \log V)$

ALGORITMI BASATI SULLA TECNICA LABEL CORRECTING

- CONSIDERERERMO ADESSO IL CASO **PIÙ GENERALE**: POSSIBILE PRESENZA DI:
  - CICLI
  - ARCHI DI PESO NEGATIVO (CIOÈ $w: E \to \mathbb{R}$)

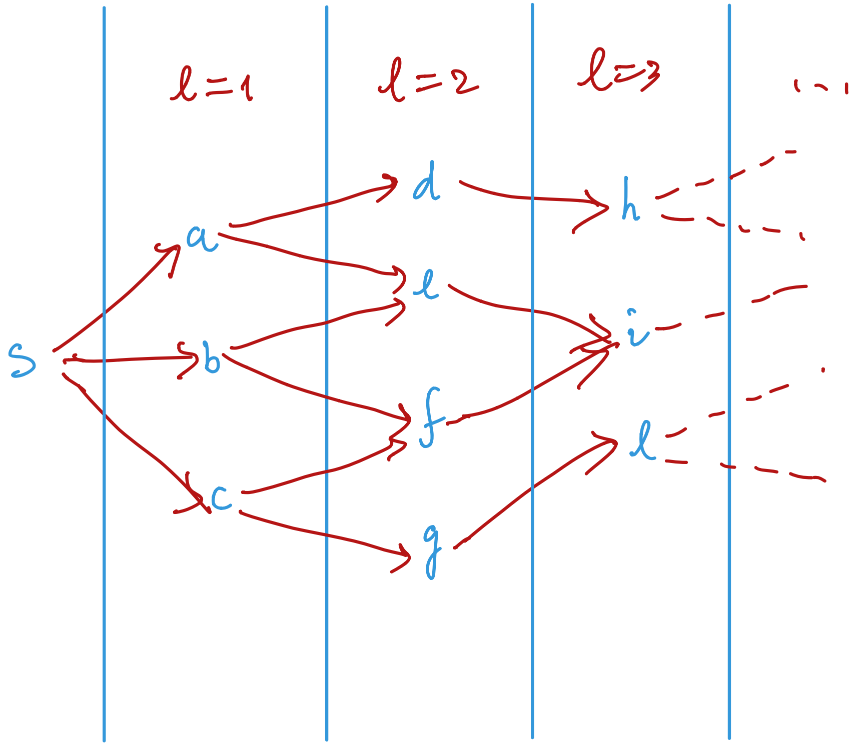
- IN TAL CASO **NON** CI SONO EURISTICHE EFFICIENTI PER CONOSCERE L'INSIEME S NEL CORSO DELLA COMPUTAZIONE

- PRESENTEREMO UN ALGORITMO (BELLMAN-FORD) BASATO SULLA TECNICA LABEL CORRECTING

- SVILUPPEREMO LA PRIMA PARTE DELL'ALGORITMO COME SE IL GRAFO **AMMETTESSE** CAMMINI MINIMI DALLA SORGENTE (CIOE' COME SE **NON CI FOSSERO** CICLI DI PESO NEGATIVO RAGGIUNGIBILI DALLA SORGENTE)

CAMMINI MINIMI E DI LUNGHEZZA MINIMA

SUPPONIAMO CHE I NODI DEL NOSTRO GRAFO SIANO STRATIFICATI COSÌ, IN BASE ALLA LUNGHEZZA DEI LORO CAMMINI MINIMI DA s DI LUNGHEZZA MINIMA:



- DOPO L'ESECUZIONE DI

  INITIALIZE-SINGLE-SOURCE(G, s)

  SI HA: $d[s] = \delta_G(s,s)$

- DOPO UNA PRIMA ESECUZIONE DEL CICLO

  for $(u,v) \in E$ **do**

  RELAX$(u,v;w)$

  SI HA: $d[a] = \delta_G(s,a)$

  $d[b] = \delta_G(s,b)$

  $d[c] = \delta_G(s,c)$

- DOPO UNA SECONDA ESECUZIONE DEL CICLO

for $(u, v) \in E$ **do**

RELAX $(u, v; w)$

SI HA: $d[d] = \delta_G(s, d)$

$d[e] = \delta_G(s, e)$

$d[f] = \delta_G(s, f)$

$d[g] = \delta_G(s, g)$

- DOPO UNA TERZA ESECUZIONE DEL CICLO

<u>for</u> $(u, v) \in E$ <u>do</u>

RELAX $(u, v; w)$

SI HA: $d[h] = \delta_G(s, h)$

$d[i] = \delta_G(s, i)$

$d[l] = \delta_G(s, l)$

- ECC.

- QUINDI DOPO $|V|-1$ ESECUZIONI DEL CICLO

  for $(u,v) \in E$ **do**

  $\quad$ RELAX $(u, v; w)$

SI HA: $d[v] = \delta(s,v)$ PER OGNI $v \in V$

<u>PURCHE' IL GRAFO $G=(V,E)$ CON FUNZIONE PESO $w: E \to \mathbb{R}$ NON CONTENGA CICLI DI PESO NEGATIVO RAGGIUNGIBILI DA s</u>

E SE (G, w) CONTENESSE CICLI DI PESO NEGATIVO RAGGIUNGIBILI DA s ?

ALLORA UN' ULTERIORE ESECUZIONE DEL CICLO

for $(u, v) \in E$ **do**

RELAX $(u, v ; w)$

AGGIORNEREBBE QUALCHE $d[u]$!

procedure BELLMAN-FORD(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)

for $i = 1$ **to** $|V| - 1$ **do**

  for $(u, v) \in E$ **do**

    RELAX$(u, v; w)$

for $(u, v) \in E$ **do**

  if $d[v] > d[u] + w(u, v)$ **then**

    return FALSE

return TRUE

COMPLESSITA'

procedure BELLMAN-FORD (G, w, s)

INITIALIZE-SINGLE-SOURCE (G, s) $\quad O(V)$

for $i = 1$ to $|V| - 1$ do

for $(u, v) \in E$ do

RELAX $(u, v; w)$ $\quad O(V+E)$ $\quad O(V \cdot (V+E))$

for $(u, v) \in E$ do

if $d[v] > d[u] + w(u, v)$ then

return FALSE $\quad O(V+E)$

return TRUE

$O(VE)$