APA Modulo 1 Lezione 7

Elena Zucca

23 marzo 2020

Ordinamento topologico

- in un grafo orientato aciclico (DAG) la relazione sull'insieme dei nodi "v è raggiungibile da u" è un ordine parziale
- un ordine totale che "raffina" questo ordine parziale si chiama ordine topologico

Definizione

Un ordine topologico su un grafo orientato aciclico $G = (V, E)$ è un ordine totale (stretto) $<$ su V tale che, per ogni $u, v \in V$:

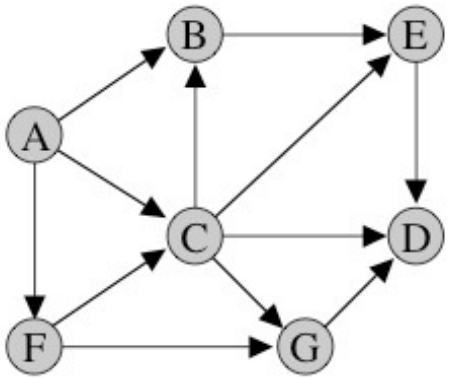
$$(u, v) \in E \Rightarrow u < v$$

Ripasso

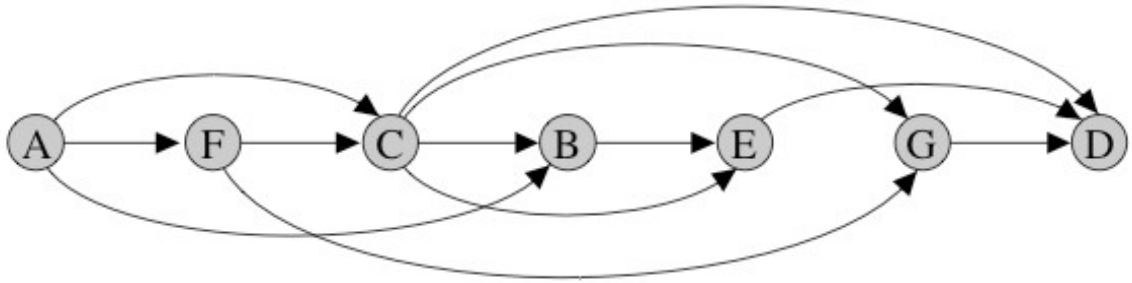
- ordine parziale $\leq$: relazione
  - riflessiva $x \leq x$
  - antisimmetrica $x \leq y$ e $y \leq x$ implica $x = y$
  - transitiva $x \leq y$ e $y \leq z$ implica $x \leq z$

  dati due elementi, non è detto che questi siano in relazione, esempio: ereditarietà in linguaggi object-oriented
- versione stretta: $x < y$ se $x \leq y$ e $x \neq y$ (si può partire da questa)
- ordine totale: come sopra, ma vale sempre $x \leq y$ o $y \leq x$
  esempio: ordinamento su numeri
- anche qui versione stretta come sopra

Problema: dato DAG trovare un ordine topologico



(a)



(b)

(a) un grafo aciclico **(b) ordine topologico dei nodi del grafo (a)**

problema rilevante per scheduling di job, realizzazione di diagramma PERT delle attività, ordine di valutazione delle caselle in un foglio di calcolo, ordine delle attività specificate da un makefile, etc.

si vede facilmente che possono esistere diversi ordini topologici per lo stesso grafo

Esiste sempre?

Definizione

In un grafo orientato un nodo è sorgente (source) se non ha archi entranti, pozzo (sink) se non ha archi uscenti.

- in un DAG esistono sempre almeno un nodo pozzo e un nodo sorgente
- per assurdo: sia x_0 un nodo, se non esiste un nodo sorgente x_0 ha almeno un arco entrante, sia (x_1, x_0)
- analogamente per x_1 si ha sia (x_2, x_1), (x_1, x_0)
- ...
- $\ldots, (x_{i+1}, x_i), \ldots, (x_1, x_0)$
- dato che i nodi sono un insieme finito prima o poi trovo lo stesso nodo, contro l'ipotesi di aciclicità
- quindi, esiste sempre un ordine topologico: esiste sempre un nodo sorgente, eliminando questo con tutti i suoi archi uscenti esiste sempre un nodo sorgente nel grafo restante, e così via

Ne segue un ovvio algoritmo

- per evitare di modificare il grafo e trovare in tempo costante, a ogni passo, un nodo sorgente, memorizziamo per ogni nodo il suo indegree
- invece di rimuovere (u, v) si decrementa il valore di indegree per v
- quando il valore di indegree per un nodo diventa zero, lo si inserisce in un insieme di nodi sorgente da cui si estrae ogni volta il nodo successivo da inserire nell'ordine topologico

Pseudocodice

```
topologicalsort(G)
  S = insieme vuoto; Ord = sequenza vuota
  for each (u nodo in G)
    indegree[u]= indegree di u
  for each (u nodo in G)
     if (indegree[u] = 0) S.add(u)
  while (S non vuoto)
    u = S.remove()
    Ord.add(u) //aggiunge in fondo
    for each ((u,v) arco in G)
      indegree[v]--
      if (indegree[v]=0) S.add(v)
  return Ord
```

Complessità

```
topologicalsort(G)
  S = insieme vuoto; Ord = sequenza vuota
  for each (u nodo in G)
    indegree[u]= indegree di u//m passi
  for each (u nodo in G)
     if (indegree[u] = 0) S.add(u)//n passi
  while (S non vuoto)//in tutto m+n passi
    u = S.remove()
    Ord.add(u)
    for each ((u,v) arco in G)
      indegree[v]--
      if (indegree[v]=0) S.add(v)
  return Ord
```

la complessità è quindi $O(n+m)$ (visita), sempre assumendo liste di adiacenza

Algoritmo alternativo

- basato su visita in profondità
- usa timestamp che indicano esplicitamente tempi di inizio e fine visita
- per semplicità assumiamo un contatore `time` globale

Pseudocodice

```
DFS(G)
  for each (u nodo in G)
    marca u come bianco; parent[u]=null
  time = 0
  for each (u nodo in G) if (u bianco) DFS(G,u)

DFS(G,u,T)
  time++; start[u] = time //inizio visita
  visita u; marca u come grigio
  for each ((u,v) arco in G)
    if (v bianco)
      parent[v]=u
      DFS(G,v)
  time++; end[u] = time //marca u come nero
  //fine visita
```

Come calcola ordine topologico?

Proprietà della visita

se G è aciclico, per ogni (u, v), in qualunque visita in profondità di G si ha:

$$end[v] < end[u]$$

Prova (semi-formale):

- se inizia prima la visita di u: durante la visita di u trovo l'arco (u, v) e v è bianco, quindi viene effettuata la chiamata `DFS(G,v)`, ossia inizia la visita di v, quindi dovrà essere necessariamente $\texttt{end}[v] < \texttt{end}[u]$
- se inizia prima la visita di v: durante la visita di v non posso trovare (un cammino da v a) u, essendo G aciclico, quindi completo la visita di v prima di iniziare quella di u, quindi a maggior ragione $\texttt{end}[v] < \texttt{end}[u]$

Quindi:

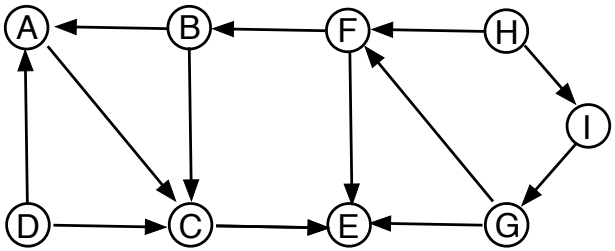
- dato che per ogni (u, v) si ha:

$$end[v] < end[u]$$

- l'ordine dei nodi di G secondo il tempo di fine visita in una visita in profondità è l'inverso di un ordine topologico
- quindi ottenere un ordine topologico dei nodi di un DAG è sufficiente effettuare una visita in profondità, inserendo a ogni fine visita il nodo in una sequenza ordinata
- la complessità è semplicemente quella della visita, quindi $O(n + m)$

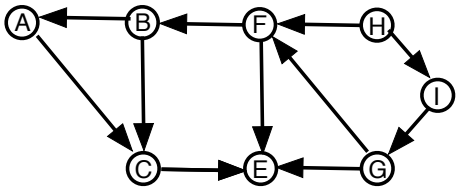
Esempio

Eseguiamo sul seguente grafo

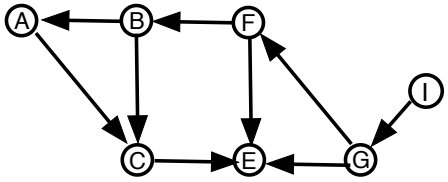


i due algoritmi di ordinamento topologico, spiegando concisamente i passi

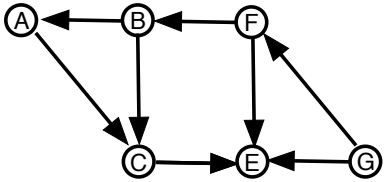
Primo algoritmo



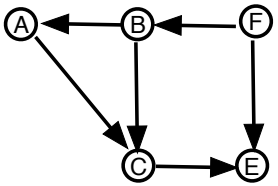
D



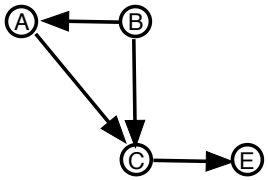
D, H



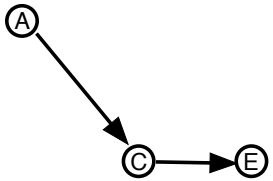
D, H, I



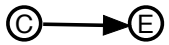
D, H, I, G



D, H, I, G, F



D, H, I, G, F, B



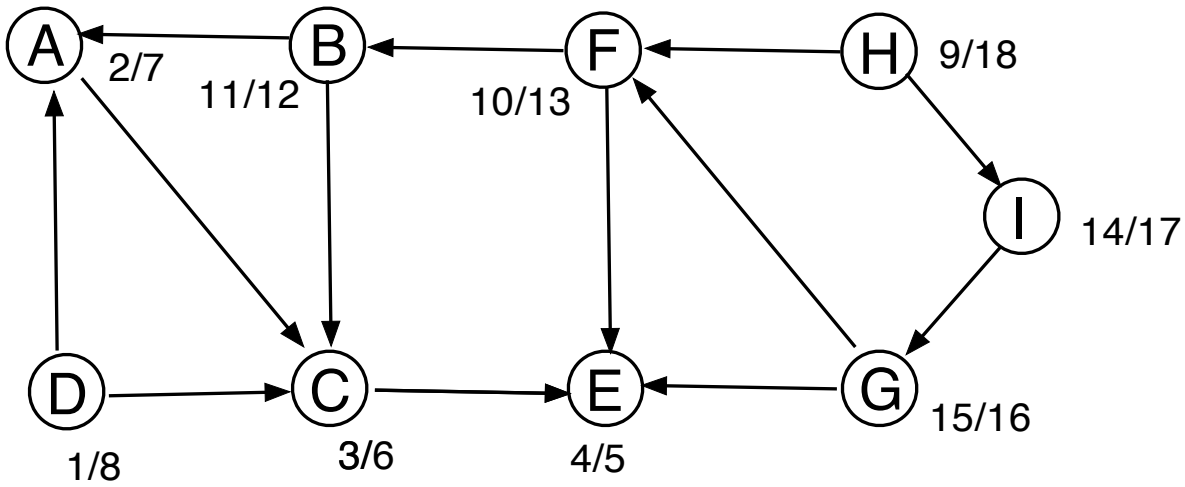
D, H, I, G, F, B, A

D, H, I, G, F, B, A, C

D, H, I, G, F, B, A, C, E

Secondo algoritmo

iniziamo per esempio la visita da D



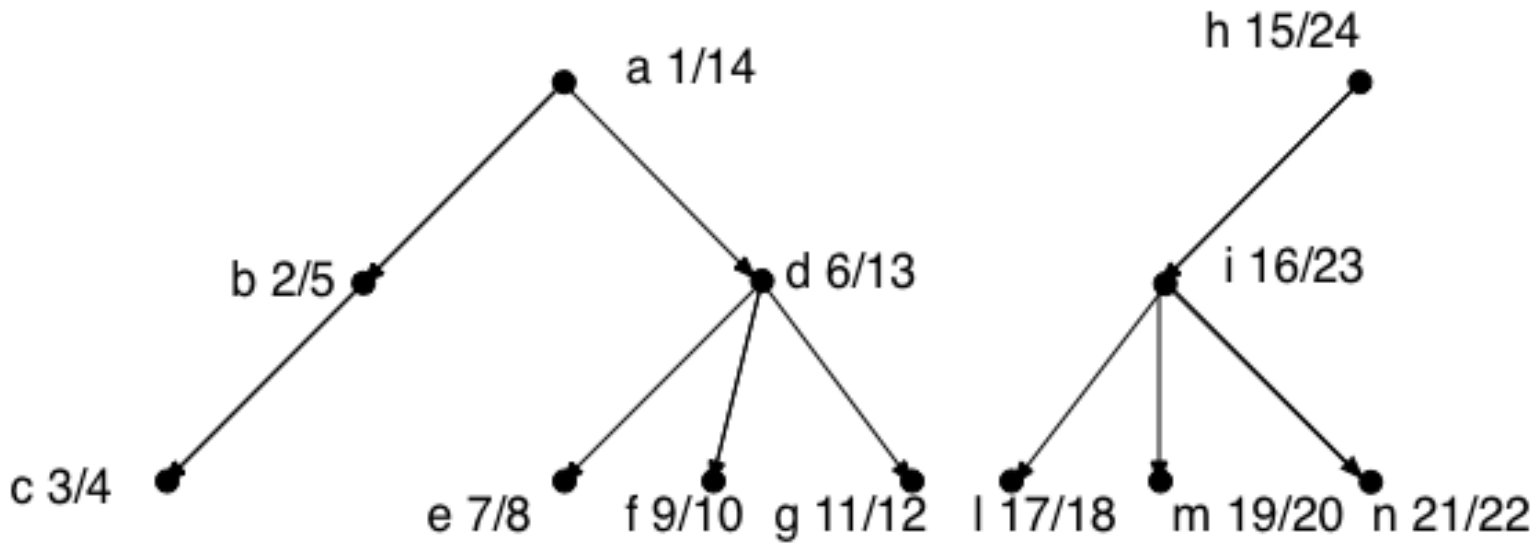
H, I, G, F, B, D, A, C,E

Un altro esercizio

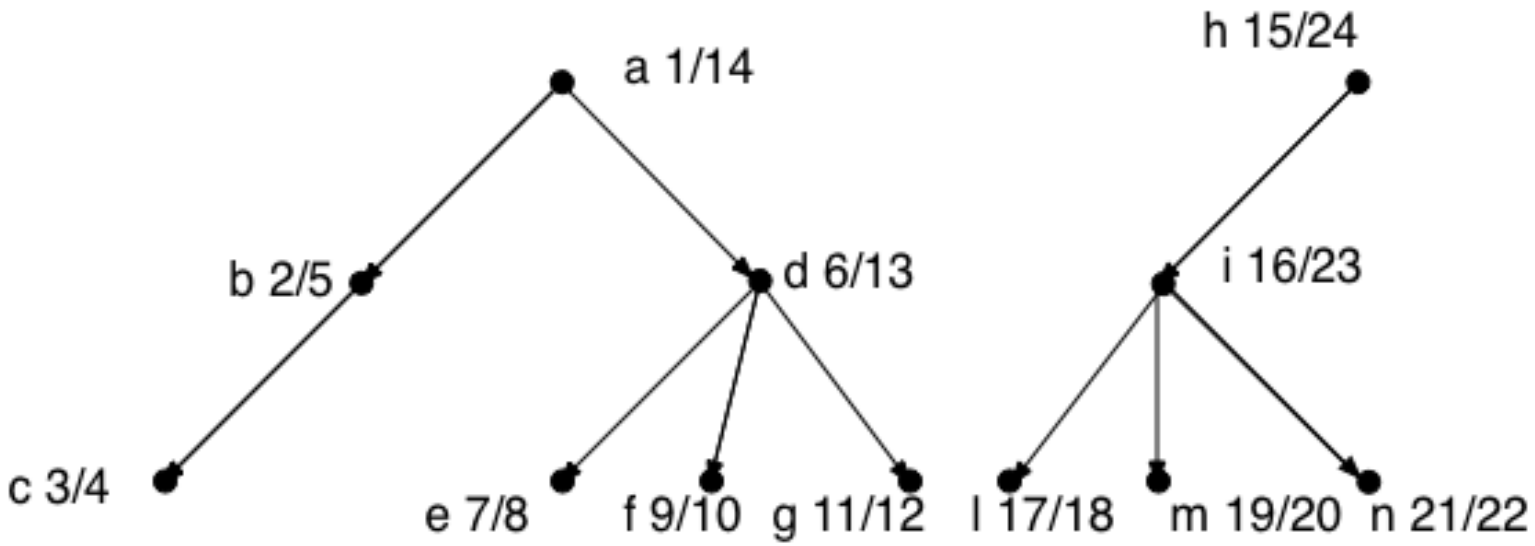
- grafo orientato G ha nodi a, b, c, d, e, f, g, h, i, l, m, n
- non ne conosciamo gli archi, ma sappiamo che la sua visita in profondità produce i seguenti timestamp:
  a 1/14, b 2/5, c 3/4, d 6/13, e 7/8, f 9/10, g 11/12, h 15/24, i 16/23
  l 17/18, m 19/20, n 21/22

1. Si disegni la foresta DFS ottenuta attraverso la visita.
2. Per ognuno dei seguenti archi si dica se può essere presente nel grafo visitato G. Si disegni un possibile G che contiene tutti gli archi per cui la risposta è positiva, mentre per quelli per cui la risposta è negativa si spieghi perché.
   - d $\rightarrow$ i
   - e $\rightarrow$ c
   - i $\rightarrow$ d
   - l $\rightarrow$ h
3. Può G contenere un ciclo di lunghezza 7?

- timestamp:
  a 1/14, b 2/5, c 3/4, d 6/13, e 7/8, f 9/10, g 11/12, h 15/24, i 16/23
  l 17/18, m 19/20, n 21/22
- foresta DFS:

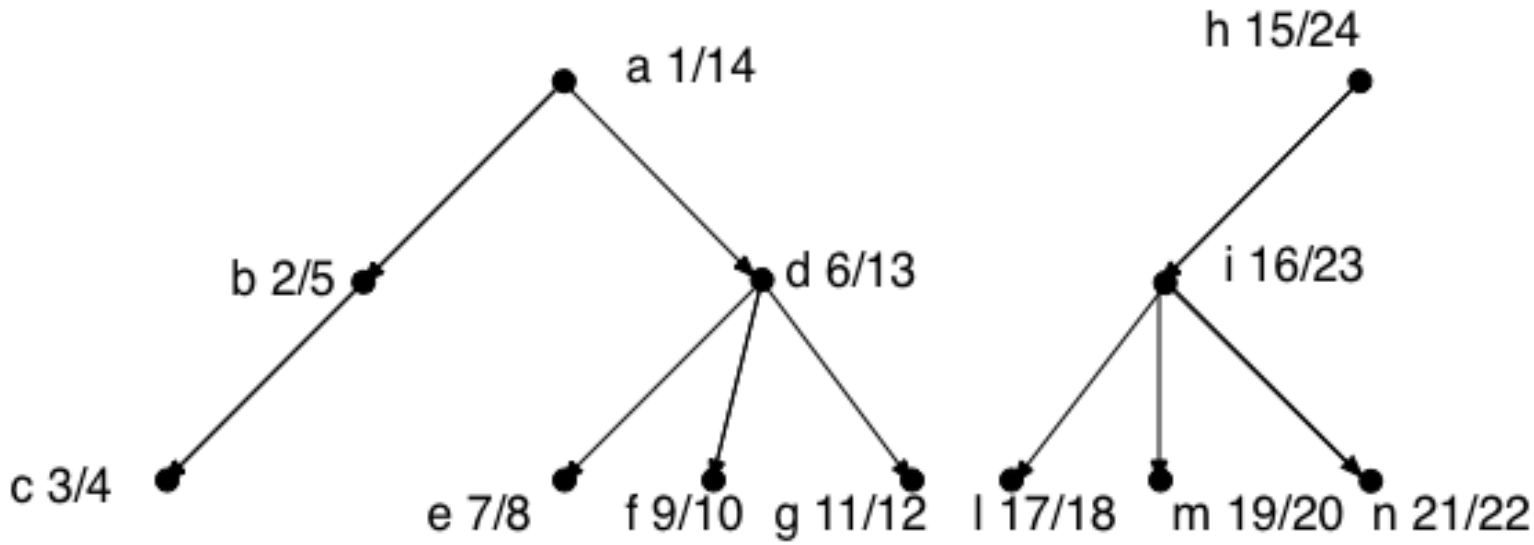


- d $\rightarrow$ i

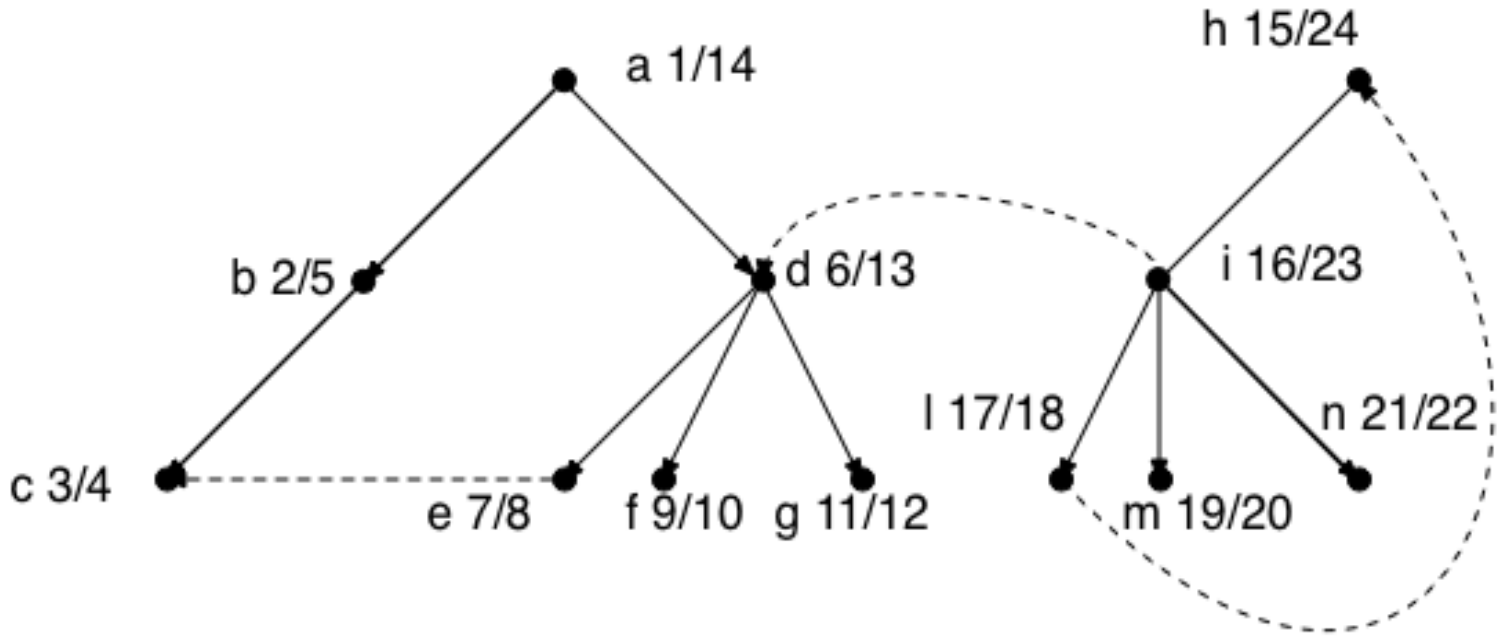


NO: la visita del nodo d, che inizia quando i non è ancora visitato, dovrebbe includere la visita di i, quindi terminare dopo quella di i

- e $\rightarrow$ c
- i $\rightarrow$ d
- l $\rightarrow$ h



- e $\rightarrow$ c
- l $\rightarrow$ h
- i $\rightarrow$ d



SI: la visita del nodo destinazione è già iniziata (il nodo è nero nei primi due casi, grigio nel terzo)