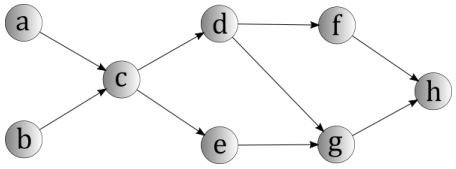
Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2023/24)

Prova scritta 30 gennaio 2025

Esercizio 1 Si consideri il seguente grafo:



1. Si ottenga un ordine topologico con il primo algoritmo visto a lezione, specificando, per ogni iterazione, l'insieme di nodi sorgente (`S`), il nodo estratto (`u`), e l'ordine corrente (`Ord`). **Ogni volta che si hanno più scelte possibili, si segua l'ordine alfabetico.**
2. Si ottenga un ordine topologico con il secondo algoritmo visto a lezione (DFS ccon timestamp), spiegando cosa succede a ogni iterazione. **Ogni volta che si hanno più scelte possibili, si segua l'ordine alfabetico** (quindi, la prima visita inizierà dal nodo a).
3. È possibile dare un ulteriore ordine topologico diverso da quelli ottenuti precedentemente?
4. È possibile, aggiungendo solo un arco, ottenere due componenti fortemente connesse?

Soluzione

1. Il primo algoritmo estrae di volta in volta dal grafo un nodo sorgente. Si ha quindi:

insieme nodi sorgente	nodo estratto	ordine
a,b	a	a
b	b	a,b
c	c	a,b,c
d,e	d	a,b,c,d
e,f	e	a,b,c,d,e
f,g	f	a,b,c,d,e,f
g	g	a,b,c,d,e,f,g
h	h	a,b,c,d,e,f,g,h

2. Il secondo algoritmo consiste nell'effettuare una visita in profondità con timestamp e poi prendere i nodi in ordine inverso di fine visita. Si ha quindi:

nodo	inizio visita	fine visita
a	1	14
c	2	13
d	3	10
f	4	7
h	5	6
g	8	9
e	11	12
b	15	16

L'ordine topologico risultante è quindi: b a c e d g f h

3. Sì, un altro ordine topologico è per esempio: b a c d e g f h
4. Sì, basta aggiungere per esempio l'arco h $\rightarrow$ b; in questo modo si ha $\{a\} \rightarrow \{b, c, d, e, f, g, h\}$.

Esercizio 2 Si consideri il seguente algoritmo ricorsivo, dove $A[0..n-1]$ è un array di interi, ordinato in senso non decrescente.

```
d(A)
  return d(A,0,n-1)
d(A,inf,sup)
  if inf ≥ sup return false
  mid = (inf+sup)/2
  if (A[mid]=A[mid+1]) return true
  return d(A,inf,mid) || d(A,mid+1,sup)
```

1. Scrivere e risolvere la relazione di ricorrenza che descrive il costo di `d` in funzione di `n`.
2. Cosa calcola `d(A)`? Giustificare la risposta, spiegando su quale principio si basa la correttezza.
3. Dire se l'algoritmo è ottimo.

Soluzione

1. La relazione di ricorrenza è la seguente (conviene esprimere n come 2^k):

   $T(2^0) = 1$
   $T(2^k) = 1 + 2\,T(2^{k-1})$, per $k > 0$.

   Utilizzando la tecnica per sostituzioni successive si ha:

   $$T(2^k) = 1 + 2 \cdot T(2^{k-1}) = 1 + 2 + 2 \cdot T(2^{k-2}) = \ldots = 2^0 + 2^1 + \ldots + 2^k = \tfrac{2^{k+1}-1}{2-1}$$

   quindi $T(n) = O(n)$.

2. L'algoritmo controlla se l'array contiene oppure no elementi duplicati. Infatti, possiamo ragionare per induzione forte sul numero di elementi. Nel caso di nessun elemento o un elemento solo l'algoritmo restituisce falso correttamente. Nel caso di almeno due elementi, la porzione di array viene divisa in due parti. Se l'ultimo elemento della prima parte è uguale al primo della seconda l'array contiene duplicati, e l'algoritmo restituisce vero correttamente. Altrimenti, dato che l'array è ordinato, vi sono duplicati solo se una delle due parti contiene duplicati, come correttamente calcolato dalle due chiamate ricorsive per ipotesi induttiva.
3. Non è possibile dare un algoritmo più efficiente, ossia il problema ha complessità $\Omega(n)$, in quanto è sicuramente necessario esaminare tutti gli elementi.

Esercizio 3 Dopo aver fatto tutte le assunzioni necessarie, dimostrare che l'EIG è in grado di trovare un accordo tra quattro processi uno dei quali, senza mai tradirsi, è in grado di spedire bit che infliggono il massimo danno possibile.