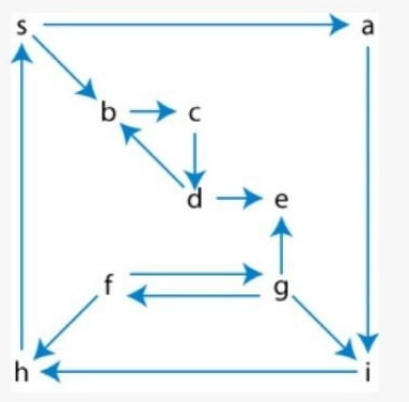
Analisi e progettazione di algoritmi

(III anno Laurea Triennale - a.a. 2024/25)

Prova scritta 3 settembre 2025

Esercizio 1 Si esegua, sul seguente grafo:



l'algoritmo per il calcolo delle componenti fortemente connesse. In particolare, si diano:

1. i tempi di inizio e fine visita ottenuti per ogni nodo in seguito a una visita in profondità nella quale, in tutti i casi in cui si deve scegliere un nodo, **si consideri l'ordine alfabetico crescente**)
2. la sequenza delle componenti fortemente connesse $\texttt{Ord}^{\leftrightarrow}$ ottenuta
3. il grafo quoziente.
4. Sappiamo che in **qualunque** visita in profondità il nodo con il massimo tempo di fine visita appartiene a una componente fortemente connessa sorgente. È vero anche che in **qualunque** visita in profondità il minimo tempo di fine visita appartiene a una componente fortemente connessa pozzo? Si giustifichi la risposta, eventualmente utilizzando il grafo precedente.

Soluzione

1. La visita in profondità assegna i seguenti tempi di inizio e fine visita:

   a 1/16 b 5/12 c 6/11 d 7/10 e 8/9 f 17/20 g 18/19 h 3/14 i 2/15 s 4/13

2. La sequenza delle componenti fortemente connesse ottenuta è la seguente:
   $\mathcal{C}_1 = \{f, g\}, \mathcal{C}_2 = \{a, s, h, i\}, \mathcal{C}_3 = \{b, d, c\}, \mathcal{C}_4 = \{e\}$.

3. Il grafo quoziente è il seguente:

   $\mathcal{C}_1 \longrightarrow \mathcal{C}_2$, $\mathcal{C}_1 \longrightarrow \mathcal{C}_4$, $\mathcal{C}_2 \longrightarrow \mathcal{C}_3$, $\mathcal{C}_3 \longrightarrow \mathcal{C}_4$

4. Non è vero. Basta considerare una visita che parta dal nodo c, quindi:

   c 1/8 d 2/7 b 3/4 e 5/6 ...

Esercizio 2) Consideriamo il problema di trovare il minimo e il massimo in un array $A[1..n]$. Un ovvio algoritmo, che richiede $2(n-1) = 2n-2$ confronti tra elementi, consiste nell'effettuare una prima scansione dell'array per trovare il minimo, e una seconda per trovare il massimo. Una soluzione migliore è un algoritmo divide-et-impera che divide l'array a metà e utilizza i risultati (coppie minimo/massimo) dei due sottoproblemi.

1. Si descriva in pseudocodice l'algoritmo divide-et-impera suggerito sopra.
2. Si calcoli il numero di confronti tra elementi richiesto da questo algoritmo.

Soluzione

1. Una descrizione dell'algoritmo in pseudocodice è la seguente.

```
minmax(A)//A[1..n]
  minmax(A,1,n)

minmax(A,inf,sup)//inf<=sup
  if (inf=sup) return (A[inf],A[sup])
  if (inf=sup-1)
    if (A[inf]<=A[sup]) return (A[inf],A[sup]); return (A[sup],A[inf])
  mid = (inf+sup)/2
  let (min1,max1) = minmax(A,inf,mid); (min2,max2) = minmax(A,mid+1,sup)
  in return (min(min1,min2), max(max1,max2))
```

2. Il numero di confronti $N(n)$ è definito dalla seguente relazione di ricorrenza (poniamo per comodità $n = 2^k$):

   $N(2^1) = 1$
   $N(2^k) = 2N(2^{k-1}) + 2$ per $k > 1$

   Procedendo per sostituzioni successive si ha:

   $N(2^k) = 2[2N(2^{k-2}) + 2] + 2 =$
   $2^2 N(2^{k-2}) + 2^2 + 2^1 = \ldots =$
   $2^i N(2^{k-i}) + 2^i + \ldots + 2^1 = \ldots$ (per $i = (k-1)$ si ottiene $N(2^1) = 1$
   $2^{k-1}+2^{k-1}+\ldots+2^1 = 2^{k-1}+(\sum_{i=0}^{k-1} 2^i)-1 = 2^k-2+2^{k-1} = 2^{k-1}(2+1)-2 = \frac{3}{2}n-2$

   Si noti che se non si tratta a parte il caso `inf=sup-1` si effettuano 2 confronti anziché 1 nel caso di due elementi, e in questo caso si ottengono 2n-2 confronti come nell'algoritmo ovvio.

Esercizio 3 Vuoi sincerarti che la tua stringa di $\ell = 10bit$ sia uguale alla copia in *cloud*. La tua stringa è 1001011011 mentre quella in *cloud* è 1001111101. Per quali numeri primi l'algoritmo *MCStringEqualityVerifier* fallirebbe? Come potresti stimare la probabilità di fallimento?

Guida alla correzione

1.2 5 a chi ha dato le CFC in ordine errato; 7,5 a chi ha dato ordine interno alle componenti errato

1.4 5 a chi fornisce almeno risposta giusta; 7,5 controesempio errato

2 $\geq$ 5 a chi ha almeno capito cosa si doveva fare (algoritmo ricorsivo divide-et-impera che restituisce coppie; array ovviamente non ordinato)