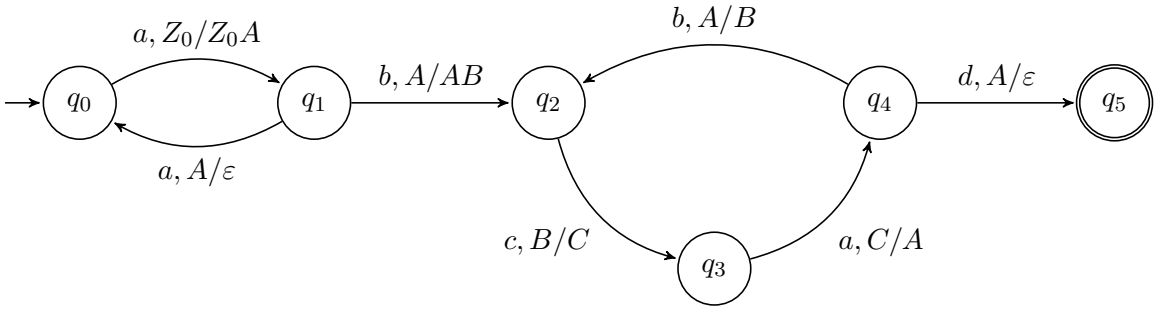
Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

17 gennaio 2024

Informatica teorica

Esercizio 1 (8 punti)
Dato il seguente automa a pila:



Si descriva sinteticamente il linguaggio da esso riconosciuto e se ne fornisca una grammatica a potenza minima che lo genera.

Soluzione

L'automa genera il linguaggio regolare L definito dalla seguente espressione regolare $a(aa)^*(bca)^+d$. È quindi sufficiente una grammatica regolare per descriverlo:

$$G = \begin{cases} S \to aA \\ A \to aA' \mid bB' \\ A' \to aA \\ B \to bB' \\ B' \to cB'' \\ B'' \to aB \mid aD \\ D \to d \end{cases}$$

Esercizio 2 (8 punti)

Si considerino le funzioni g e h: $g(x) = \begin{cases} f_x(x) + 2 & \text{se } f_x(x) \neq \bot \\ 1 & \text{altrimenti} \end{cases}$ $\qquad h(x) = \begin{cases} f_x(x) & \text{se } f_x(x) \neq \bot \\ 1 & \text{altrimenti} \end{cases}$

a) g e h sono totali?

b) g è computabile?

c) h è computabile?

Soluzione

a) g e h sono chiaramente totali in quanto non sono mai indefinite per loro stessa definizione.
b) g non è, però, computabile. Questo si può vedere per riduzione, mostrando che, se g fosse computabile, allora lo sarebbe anche un'altra funzione, che è invece notoriamente non computabile.

   Consideriamo quindi la seguente funzione g':

$$g'(x) = \begin{cases} 1 & \text{se } f_x(x) \neq \bot \\ 0 & \text{altrimenti} \end{cases}$$

È noto (da lezione) che la funzione g' non è computabile. Se g fosse computabile, però, lo sarebbe anche g'. Infatti, per calcolare $g'(x)$ sarebbe sufficiente calcolare $g(x)$ e poi, se vale $g(x) \geq 2$, restituire 1, altrimenti, se vale $g(x) = 1$, restituire 0 (non ci sono altri casi possibili di valori di $g(x)$). Di conseguenza, g non può essere computabile.

c) Nemmeno h è computabile. È noto infatti (sempre da lezione) che la funzione (computabile)

$$\bar{h}(x) = \begin{cases} f_x(x) + 1 & \text{se } f_x(x) \neq \perp \\ \perp & \text{altrimenti} \end{cases}$$

non ammette alcuna estensione che sia totale **e** computabile. Si noti che la funzione $h'(x) = h(x) + 1$ è, per definizione, una estensione totale di $\bar{h}$, quindi, per quanto detto sopra, non è computabile. Tuttavia, se h fosse computabile, allora lo sarebbe anche h', portando quindi ad una contraddizione. Di conseguenza, neanche h è computabile.

Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

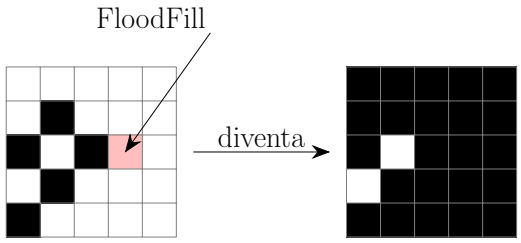
17 gennaio 2024

Algoritmi e strutture dati

Esercizio 3 (8 punti)

Si consideri il seguente algoritmo `FloodFill`, in cui l'array bidimensionale quadrato `A` descrive una figura monocromatica rappresentata come bitmap – ogni cella contiene 0 (cella vuota) o 1 (cella piena). Si vuole, in particolare, riempire la figura a partire dalla posizione (x, y) in `A`, fino a incontrare i bordi della figura o la fine di `A` (si assuma che, orizzontalmente e verticalmente, le posizioni di `A` varino da 1 a n).

```
FloodFill(A, x, y)
1. IF NOT (1 <= x <= n AND 1 <= y <= n) THEN RETURN
2. IF A[x, y] = 1 THEN RETURN
3. A[x, y] := 1
4. FloodFill(A, x-1, y)
5. FloodFill(A, x+1, y)
6. FloodFill(A, x, y-1)
7. FloodFill(A, x, y+1)
```



a) Si fornisca un limite superiore (il più stretto possibile) alla complessità asintotica di `FloodFill`.

b) Si fornisca anche un limite inferiore alla complessità asintotica di `FloodFill`.

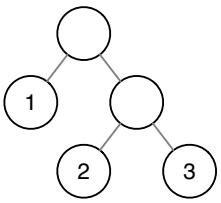
Soluzione

a) Possiamo osservare che, per una qualunque cella di coordinate (k, h), la chiamata `FloodFill(A,k,h)` può avere origine da al più 4 celle confinanti e che quindi la complessità sia dell'ordine di $\mathcal{O}(4n^2) = \mathcal{O}(n^2)$.
b) Questo è anche un limite inferiore, perché nel peggiore dei casi si devono riempire n^2 celle, eseguendo altrettante volte la riga 3. Pertanto la complessità è $\Theta(n^2)$.

Esercizio 4 (8 punti)

Si considerino alberi binari in cui i dati vengono memorizzati solo nelle foglie: in essi sono presenti due tipi di nodi, ramo e foglia. Un nodo `b` di tipo ramo serve a memorizzare i nodi interni, ed è dotato di due componenti `b.left` e `b.right`, puntatori rispettivamente al sottoalbero sinistro e destro. Un nodo `t` di tipo foglia invece contiene esclusivamente una componente `t.value` con il dato memorizzato. Si assuma che i puntatori usati nell'albero non siano mai NIL. Si supponga inoltre di avere due predicati `IsBranch(T)` e `IsLeaf(T)` per controllare se un oggetto T sia rispettivamente un ramo oppure una foglia.

Esempio di albero:



Siano $e_1, e_2, \ldots, e_n$ tutte le foglie, da sinistra a destra, presenti in un albero T; sia inoltre f una funzione con due argomenti applicabile ai dati memorizzati in T e z un valore iniziale. Si definisca una procedura `fold`$(f, z, T.root)$, che restituisca $f(e_1, f(e_2, \ldots f(e_n, z))\ldots)$ e se ne valuti la complessità temporale in funzione del numero di nodi in T.

Soluzione

```
fold(f, z, T)
   if IsLeaf(T) then return f(T.value, z)
   else
       r := fold(f, z, T.right)
       return fold(f, r, T.left)
```

In pratica si tratta di una visita dell'albero, quindi la complessità è $\Theta(N)$, dove N è il numero di nodi nell'albero.