Algoritmi e Principi dell'Informatica

Appello del 16 Luglio 2015

Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 2 ore e 30 minuti.

Chi deve sostenere solo il modulo di *Informatica teorica* deve svolgere gli Esercizi 1 e 2 in 1 ora e 15 minuti.

Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 3 e 4 in 1 ora e 15 minuti.

NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

Esercizio 1 (punti 8)

Scrivere una grammatica a potenza minima che generi il linguaggio $\{ a^n b^m c^l \mid n \neq m \text{ o } n \neq l \}$

In alternativa è possibile costruire un automa, sempre a potenza minima, che riconosca lo stesso linguaggio; ciò però comporta una diminuzione del punteggio ottenibile di 2 punti.

Esercizio 2 (punti 8)

Parte a

È decidibile stabilire se una macchina di Turing M entra più di una volta in un qualche stato dell'organo di controllo quando riceve in input una stringa w?

In altre parole:

Indicata con <M> la codifica di una MT M e con <w> quella di una stringa sull'alfabeto di input di M, si dica se è computabile la funzione

F(<M>, <w>) = 1, se quando M opera su w esistono due configurazioni di M
con lo stesso stato dell'organo di controllo
0 altrimenti

Parte b

È decidibile stabilire se una macchina di Turing M (con un unico stato iniziale dell'organo di controllo) entra più di una volta nello stato iniziale dell'organo di controllo quando riceve in input una stringa w?

In altre parole:

Indicata con <M> la codifica di una MT M, con <w> quella di una stringa sull'alfabeto di input di M e con q_0 lo stato iniziale dell'organo di controllo di M, si dica se è computabile la funzione

F(<M>, <w>) = 1, se quando M opera su w esistono due configurazioni di M
in cui lo stato dell'organo di controllo è q_0
0 altrimenti

Giustificare brevemente le risposte date.

Esercizio 3 (punti 10)

Dato un albero binario di ricerca T, i cui nodi memorizzano una variabile che ne rappresenta il colore, si vuole determinare se T è un albero Rosso-Nero. Si definisca un algoritmo che risolva questo problema e se ne discuta la complessità.

Esercizio 4 (punti 5)

Parte a

Si consideri una tabella hash, con m = 10 posizioni, le cui chiavi sono interi non negativi. La tabella è gestita mediante indirizzamento aperto.

Data la funzione hash:

$h(k,i) = (k + 2 \times i) \bmod m$

si discuta la bontà di *h(k,i)* rispetto alle sequenze di ispezione che essa genera nella tabella considerata.

Parte b

Si consideri una generica tabella hash con m posizioni e fattore di carico $\alpha = 1/2$.
Si considerino i due casi di tabella con liste concatenate e di tabella aperta: qual è, nel caso pessimo, il tempo necessario (è sufficiente l'ordine di grandezza) per la ricerca e l'inserimento di un elemento in tabella nei due casi? Giustificare brevemente la risposta.

Soluzioni schematiche

Esercizio 1

S -> A | B *(A è il sottolinguaggio con n≠m, B quello con n≠l)*

A -> A1 A2 C0 | A2 B1 C0 *(sottolinguaggio con n≠m. due modi per farlo:*

1) ho più a che b e quindi ho un prefisso non vuoto di a (A1) seguito da a e b bilanciate (A2), seguite da c arbitrarie (C0)

2) ho meno a che b e quindi ho a e b bilanciate (A2) seguite da un prefisso non vuoto di b (B1) seguito da c arbitrarie (C0))

A2 -> a A2 b | ε

B -> A1 B2 | B2 C1 *(sottolinguaggio con n≠l. due modi per farlo:*

1) ho più a che c e quindi ho un prefisso non vuoto di a (A1) seguito da a e c bilanciate con in mezzo delle b arbitrarie (B2)

2) ho meno a che c e quindi ho a e c bilanciate con in mezzo delle b arbitrarie (B2) seguite da un prefisso non vuoto di c (C1))

B2 -> aB2c | B0 *(a e c bilanciate con in mezzo delle b arbitrarie (B0))*

A1 -> a A1 | a

B1 -> b B1 | b

C1 -> c C1 | c

B0 -> B1 | ε

C0 -> C1 | ε

Esercizio 2

Parte a

Il problema è decidibile. Mentre le possibili configurazioni di una MT sono infinite, gli stati dell'organo di controllo sono finiti (indichiamoli con Q_M). Basta infatti simulare al più $|Q_M| + 1$ passi dell'esecuzione di M su w e verificare se uno stato è visitato almeno due volte (NB: l'esecuzione di M su w potrebbe non terminare, e potrebbe anche darsi che la testina di lettura di M non legga alcun carattere di w).

Parte b

Il problema non è decidibile. Infatti si può ridurre ad esso il problema dell'arresto di una TM nel modo seguente: data M,

- costruisco una la macchina M' ottenuta aggiungendo a M un nuovo stato iniziale q_0', e una transizione iniziale che porta da q_0' a q_0 senza fare nient'altro; aggiungo poi delle transizioni che da ogni stato finale di M portano a q_0' e fermano la macchina; l'unico stato finale di M' è q_0'. Chiaramente M e M' riconoscono lo stesso linguaggio e M accetta una stringa, ossia si ferma, se e solo se M' raggiunge il suo stato iniziale più di una volta.

Esercizio 3

L'algoritmo seguente verifica che le proprietà degli alberi Rosso-Neri siano rispettate: verifica che la radice sia nera, quindi richiama una funzione che ricorsivamente analizza i sottolaberi per calcolarne l'altezza nera e controllare che i figli dei nodi rossi siano neri. La complessità è proporzionale al numero di nodi dell'albero.

```
checkRB (Tree T) {
 if (T.color=RED)
     return FALSE
 (bh, RB) <-- checkSubTree(T, 0); /*bh = altezza nera del
                        sottoalbero T; RB = valore di verità
                         per le proprietà RB */

 return RB;
}
checkSubTree(Tree T, int bh) {
     if (T=NIL)
          return (bh, TRUE)
     if  ((T.color  =  RED)  AND  (T.left.color  =  RED  OR
            T.right.color = RED))
          return (bh, FALSE)
     if (T.color = BLACK)
          bh = bh+1

     (bh_left, RBleft) = checkSubTree(T.left, bh)
     (bh_right, RBright) = checkSubTree(T.right, bh)

     if not (RBleft AND RBright)
          return (bh, FALSE)
     if (bh_left != bh_right)
          return (bh, FALSE)

     return (bh_left, TRUE)
}
```

Esercizio 4

Parte a

Visto che m è pari, le sequenze di ispezione generate visitano solo le posizioni pari o solo quelle dispari (dipende dalla prima posizione visitata). Se m fosse dispari, la funzione permetterebbe invece di ispezionare tutti gli elementi della tabella.

Parte b

In entrambi i tipi di tabella il caso pessimo produce n/2 conflitti sulla stessa posizione determinando quindi un tempo di ricerca O(n) in ambo i tipi di tabella; se si vogliono evitare occorrenze ripetute è necessario comunque anche scandire l'intera lista sia nel caso di tabella aperta che di tabella con liste concatenate. Quindi il tempo di ricerca e inserimento nel caso pessimo è comunque O(n).