Algoritmi e Principi dell'Informatica
Seconda Prova in Itinere

8 Febbraio 2013

Il tempo a disposizione è di **2 ore**

Esercizio 1 (punti 5/15-esimi)

Esercizio 1.a

Si delinei una macchina di Turing M a *nastro singolo* che riconosca il linguaggio $L \subseteq \{a,b\}^*$ consistente di tutte e sole le stringe con un numero di occorrenze di 'a' uguale a quello di 'b'. La descrizione non deve necessariamente scendere in tutti i dettagli ma deve essere sufficientemente chiara e precisa da permettere di calcolare la complessità temporale asintotica della macchina: si spieghi con precisione quale sia tale complessità.

E' possibile trovare un altro modello di calcolo a potenza risolutiva minore della macchina a nastro singolo (ossia non in grado di riconoscere tutti i linguaggi riconoscibili dalla MT) che riconosca L con complessità inferiore a quella di M? Si motivi brevemente la risposta.

Esercizio 1.b

Si risolva lo stesso esercizio del punto 1.a con riferimento al linguaggio $L' \subseteq \{a, b, c\}^*$ le cui stringhe hanno ugual numero di 'a', 'b' e 'c'.

Esercizio 2 (punti 8/15-esimi)

Si consideri un array che sia il risultato della visita in post-ordine sinistro di un Binary Search Tree (BST) i cui nodi contengano numeri interi ordinati in ordine crescente (figlio di sinistra < del padre ≤ del figlio di destra; linearizzato in: figlio di sinistra, figlio di destra, padre).

Assumendo l'ipotesi che il BST di partenza avesse entrambi i figli per ogni nodo interno, si delinei, mediante opportuno pseudocodice, un algoritmo che, partendo dall'array linearizzato ricostruisca il BST originario. Indi se ne valuti la complessità asintotica.

NB: il punteggio attribuito alla soluzione terrà conto della complessità asintotica dell'algoritmo.

Esercizio 3 (punti 3/15-esimi)

Si vuole progettare una tabella di hashing in cui memorizzare numeri primi, usando gli stessi come chiave. La tabella è lunga 2^n elementi, si sceglie di utilizzare la funzione $h(x) = x \textbf{ mod } 2^n$ come funzione di hash. E' una buona scelta? Motivare brevemente la risposta.

Esercizio 4 (punti 2/15-esimi)

Si stabilisca l'ordine di grandezza (classe di Θ-equivalenza) della soluzione della seguente equazione alle ricorrenze:

$T(n) = 7\ T(n/5) + \Theta\ (n^2)$

Tracce di soluzioni

Esercizio 1

1.a

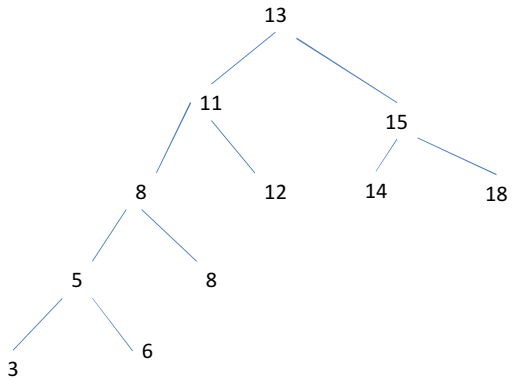
Una M a nastro singolo non può far altro che scorrere l'intero nastro, o una sua porzione ad esso proporzionale, nel caso pessimo, per ogni carattere letto alla ricerca di un suo corrispondente; alternativamente aggiornando un contatore che dovrebbe trovarsi in una diversa porzione di nastro rispetto a quella dove è memorizzata la stringa di ingresso. Quindi la sua complessità temporale deve essere almeno $\Theta(n^2)$.
Un automa a pila deterministico invece può facilmente riconoscere L in $\Theta(n)$.

1.b

La precedente macchina M può facilmente essere modificata in modo da riconoscere L' con la stessa complessità asintotica.
L' però non è riconoscibile da alcun automa a pila; è tuttavia riconoscibile da una rete di Petri, sempre in tempo lineare, che però non ha la potenza computazionale delle MT.

Esercizio 2

Un esempio di BST che soddisfi l'ipotesi dell'esercizio e della sua linearizzazione in post-ordine sinistro è fornito in figura.



(a) esempio di BST

(b) array risultante dalla visita in post-odine sinistro del BST in figura (a):

3, 6, 5, 8, 8, 12, 11, 14, 18, 15, 13.
Si noti che (per un certo valore) non possono esistere più di due nodi contenenti quel valore, altrimenti si dovrebbe violare l'ipotesi che ogni nodo interno ha entrambi i figli.

Un modo semplice di riottenere il BST di partenza consiste nel percorrere l'array da destra a sinistra ed eseguire una serie di inserimenti in ordine dei vari

elementi. In questo modo però la complessità risulta (O(n.log(n)) nel caso di BST bilanciato e $O(n^2)$ nel caso pessimo.

Un algoritmo più efficiente invece può fare uso di una struttura ausiliaria a pila e procedere nel modo seguente, da sinistra a destra, sfruttando la proprietà che gli elementi dell'array sono in ordine crescente tranne quando si passa da figlio (necessariamente di destra) a padre:

push a[0]; crea un nodo per a[0]
for i := 1, …,n
do
while a[i-1] < a[i] push a[i]; crea un nodo per a[i]
crea un nodo per a[i]; assegna al suo campo figlio di destra il puntatore al nodo creato per l'elemento in cima alla pila e al figlio di sinistra il puntatore al nodo creato per l'elemento sottostante; pop; pop; push a[i];

L'algoritmo visita ogni elemento una sola volta e quindi ha complessità O(n).

Esercizio 3
La funzione di hash proposta non è una buona funzione: i numeri primi, fatto salvo 2, sono tutti dispari; quindi i posti di posizione pari verranno utilizzati solo in caso di conflitti e la tabella risulterà fortemente sbilanciata e/o sottooccupata.

Esercizio 4
Applicando il teorema dell'esperto (Master theorem) abbiamo:
$a = 7, b = 5, f(n) = \Theta(n^2) > \Theta(n^{\log_b(a) + \varepsilon})$
Quindi la soluzione è $\Theta(n^2)$.