Algoritmi e Principi dell'Informatica

Appello del 12 Luglio 2012

Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 2 h 30'

Chi deve sostenere solo il modulo di Informatica teorica deve svolgere l'Esercizio 1 e l'Esercizio 2 in 1 ora 15'.

Chi deve sostenere solo il modulo di Informatica 3 deve svolgere l'esercizio 3 e l'esercizo 4 in 1 h 15'.

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

Esercizio 1 (punti 7/30-esimi)

Si consideri la seguente macchina di Turing M, dove q_0 è stato iniziale e q_4 finale:

$\delta(q_0,a,Z_0,Z_0) = <q_1,Z_0,Z_0,S,R,S>$

$\delta(q_1,a,\_,Z_0) = <q_1,A,Z_0,R,R,S>$

$\delta(q_1,b,\_,Z_0) = <q_2,\_,Z_0,S,L,R>$

$\delta(q_2,b,A,\_) = <q_2,A,B,R,S,R>$

$\delta(q_2,c,A,\_) = <q_3,C,\_,R,L,L>$

$\delta(q_3,c,A,B) = <q_3,C,B,R,L,S>$

$\delta(q_3,\_,Z_0,B) = <q_4,Z_0,B,S,S,S>$

Si definisca un automa a potenza minima che accetti lo stesso linguaggio di M.

Esercizio 2 (punti 9/30-esimi)

Siano $\mathcal{A}$ e $\mathcal{G}$, rispettivamente una generica famiglia di automi e una di grammatiche; siano poi $\mathcal{L}(\mathcal{A})$ e $\mathcal{L}(\mathcal{G})$ le rispettive classi di linguaggi definiti da esse. Di esse si sappia che:

- $\mathcal{L}(\mathcal{A})$ è ricorsivamente contenuta in $\mathcal{L}(\mathcal{G})$; ossia esiste un algoritmo che dato un automa A in $\mathcal{A}$ costruisce una G in $\mathcal{G}$ ad esso equivalente, ossia tale che L(A) = L(G).
- $\mathcal{L}(\mathcal{A})$ è ricorsivamente chiusa rispetto al complemento; ossia esiste un algoritmo che dato un automa A in $\mathcal{A}$ costruisce un automa A' in $\mathcal{A}$ che accetta il complemento di L(A).
- $\mathcal{L}(\mathcal{G})$ è ricorsivamente chiusa rispetto all'intersezione; ossia esiste un algoritmo che date due grammatiche G e G' in $\mathcal{G}$ costruisce una grammatica G" in $\mathcal{G}$ tale che L(G") = L(G)$\cap$ L(G').

Sulla base di queste e solo queste informazioni è possibile concludere che il segente problema:

Dati una generica G in $\mathcal{G}$ e un generico A in $\mathcal{A}$, tutte le stringhe generate da G sono accettate da A?

è decidibile?

Nel caso di risposta positiva si fornisca una breve descrizione di un algoritmo per risolvere il suddetto problema (NB: in un caso estremo si potrebbe anche dimostare la decidibilità del problema senza per questo essere in grado di produrre un tale algoritmo). E' anche opportuno fornire qualche esempio di famiglie $\mathcal{A}$ e $\mathcal{G}$ che, soddisfacendo le ipotesi di cui sopra, godano della relativa proprietà di decidibilità.

In caso di risposta negativa:

Si fornisca una convincente spiegazione per essa; un modo semplice, sempre che ciò sussista, sarebbe fornire un adeguato *controesempio*, ossia un caso specifico di $\mathcal{A}$ e $\mathcal{G}$ che soddisfano le ipotesi ma tali per cui il problema indicato non sia decidibile.

Sempre in caso di risposta negativa, è possibile individuare una nuova condizione, da aggiungere alle precedenti, che sia sufficiente per rendere decidibile il problema? Anche in questo caso si fornisca qualche giustificazione: uno schema di algoritmo, qualche esempio, …

Esercizio 3 (punti 8)

Si considerino alberi binari, in cui ciascun nodo contiene come informazione un numero intero. Si progetti un algoritmo che, ricevendo in input un albero e un intero TOT, restituisce 1 se esistono almeno due cammini distinti dalla radice alle foglie in cui la somma dei numeri contenuti nei nodi è uguale a TOT e 0 altrimenti. Due cammini sono distinti se differiscono anche solo per un nodo.

Si calcoli l'ordine di grandezza (Θ) della complessità temporale dell'algoritmo proposto in funzione del numero n di nodi dell'albero.

Esercizio 4 (punti 8)

Si consideri il linguaggio $L=\{ uu \mid u \in (a, b)^+ \}$

1. Si definisca in maniera sufficientemente precisa e dettagliata una MT *deterministica* M che riconosca L, preferibilmente con complessità temporale nella classe $\Theta(n)$, dove al solito $n = |x|$ è la lunghezza della stringa di ingresso. Si calcoli anche la funzione di complessità temporale $T_M(n)$ di M in modo preciso, non solo con riferimento alla sua classe Θ.
2. Si stabilisca se una MT *nondeterministica* N possa riconoscere lo stessso linguaggio L con una complessità temporale $T_N(n)$ tale che $T_N(n) < T_M(n)$ per ogni n. Nel caso positivo si descriva, anche a grandi linee ma in maniera sufficientemente precisa da poterne ricavare con esattezza la funzione di complessità temporale, una tale macchina N. Altrimenti si spieghi il motivo per cui una tale macchina N non può esistere[1].
3. Si delinei poi una macchina RAM che riconosca lo stesso linguaggio e se ne stimi la complessità temporale $T_R(n)$ facendo riferimento sia al criterio di costo uniforme che a quello logaritmico.

[1] Per comodità si richiama qui la Definizione 9.11. del testo in cui si definisce la funzione di complessità temporale di una MT nondeterministica.

Definizione 9.11

Sia M una MT multinastro non deterministica che accetta il linguaggio L. M ha complessità temporale T_M (e, rispettivamente, complessità spaziale S_M) con dominio I^* se, per ogni stringa di ingresso x, $x \in L$ se e solo se esiste almeno una computazione di M che accetta x in $T_M(x)$ mosse (rispettivamente, usando $S_M(x)$ celle di memoria) e nessun'altra computazione accetta x in meno di $T_M(x)$ mosse (rispettivamente, usando meno di $S_M(x)$ celle del nastro di memoria). $T_M(n)$ (rispettivamente, $S_M(n)$) viene definita come il massimo di $T_M(x)$ (rispettivamente, $S_M(x)$) su tutte le $x \in L$ di lunghezza n.

■

Tracce di Soluzioni

Esercizio 1

Il linguaggio riconosciuto dalla MT è { $a^n b^+ c^n$, n>0}

Quindi esso è riconosciuto anche da un automa a pila deterministico facilmente costruibile.

Esercizio 2

Sulla base delle sole ipotesi indicate il problema non è a priori decidibile. Ad esempio si scelga come $\mathcal{A}$ la classe degli automi a stati finiti e come $\mathcal{G}$ la classe delle grammatiche non ristrette: le due famiglie soddisfano le ipotesi; tuttavia stabilire se per una generica G in $\mathcal{G}$ L(G) $\subseteq$ L(A), A essendo un generico automa a stati finiti, è evidentemente indecidibile (altrimenti sarebbe decidibile stabilire se un generico L(G) è vuoto).

Viceversa è proprio la decidibilità della emptiness per i linguaggi L(G) che, aggiunta alle ipotesi precedenti, rende il problema decidibile. Infatti, dati G e A, L(G) $\subseteq$ L(A) se e solo se

L(G) $\cap$ $\neg$L(A) = $\emptyset$: le due operazioni insiemistiche sono effettive e la nuova ipotesi garantisce la decidibilità dell'uguaglianza.

Purtroppo, tra le famiglie di automi e grammatiche di uso più comune le uniche che soddisfano tutte queste ipotesi sono automi e grammatiche regolari, tra l'altro equipotenti tra loro;infatti è su questa classe fondamentale che si appoggiano moderni algoritmi di verifica basati su questa decisione. A meno di non ricorrere a classi decisamente più specialistiche …

Esercizio 3

```
int cerca(Tree t, int totale) {
	if(t==null)
		return 0;
	if(cercaAux(t, totale)>=2)
		return 1;
	else
		return 0;
}

int cercaAux(Tree t, int totale) {
	if(t==null)
		return 0;
	if(t->left==null && t->right==null) {
		if(totale==t->info)
			return 1;
		else
			return 0;
	} else
		return cercaAux(t->left, totale - t->info) + cercaAux(t->right, totale - t->info);
}
```

E’ facile constatare che la complessità temporale dell’algoritmo proposto è $\Theta(n)$.

Esercizio 4

Punti 1 e 2 (Impostazione)

Un MT "naturale" deterministica M che riconosca L può operare nel modo seguente:

1. Scandisce l'intera stringa x: n mosse (+ un numero limitato di eventuali mosse extra per inizializzazione e riconoscimento di fine stringa). Nel fare ciò memorizza contemporaneamente il valore di n/2 in unario in un nastro di memoria e ricopia il valore di x in un altro nastro
2. mediante n/2 mosse; riporta la testina del nastro di ingresso a metà stringa.
3. Mediante ulteriori n/2 mosse *da destra a sinistra* verifica che la prima metà della stringa nel nastro di ingresso coincida con la seconda metà della stringa nel nastro di memoria. In caso di confronto positivo la stringa è accettata.

Il totale delle mosse impiegate da M è dunque 2.n + c (c essendo una costante piccola indipendente da n).

Una MT nondeterministica N può invece "indovinare" la metà della stringa di ingresso e operare quindi nel modo seguente:

1. Scandisce e copia la stringa di ingresso in un nastro di memoria, finché nondeterministicamente "indovina" di essere giunta alla sua metà: ciò comporta quindi n/2 mosse.
2. Mediante altre n/2 mosse riporta la testina del nastro di memoria a inizio stringa.
3. Mediante ulteriori n/2 mosse verifica che la parte restante della stringa di ingresso coincida con la stringa contenuta nel nastro di memoria.

Il numero di mosse complessivo è perciò 3/2.n + c'.

Punto 3 (Impostazione)

La RAM, essendo deterministica, deve operare secondo le stesse linee di M (tenendo però presente che la definizione normale della RAM non permette alla macchina di riposizionare la testina di lettura); precisamente:

1. Legge e memorizza la stringa di ingresso, calcolandone la lunghezza mediante un contatore intero;
2. Dimezza il valore del contatore.
3. Per ogni valore di i, 1 <= i <= n/2 verifica che M[i] sia uguale a M[i+n/2].

Il costo complessivo, a criterio uniforme è dunque $k_1.n + 1 + k_2.n/2 + k_3$, ki essendo costanti piccole che indicano il numero di operazioni elementari (READ, LOAD, STORE, …) all'interno dei vari cicli.

A criterio di costo logaritmico l'esporessione precedente diventa $k_1.n.\log(n) + \log(n) + k_2.n/2.\log(n) + k_3$.