Informatica Teorica

Appello del 17 Luglio 2007

Avvisi importanti

- Il tempo disponibile per lo svolgimento della prova è **2 ore**.
- Come in altre occasioni i risultati e le successive comunicazioni (in particolare le modalità di accettazione/rifiuto del voto proposto) saranno pubblicati nella pagina personale del docente sul sito ufficiale del Dipartimento.

Esercizio 1 (punti 12)
Parte A
Si considerino le grammatiche seguenti:

G1: $S \rightarrow SS \mid ASB \mid BSA \mid \varepsilon$
$A \rightarrow a$
$B \rightarrow b$

G2: $S \rightarrow SBA \mid \varepsilon$
$BA \rightarrow AB$
$AB \rightarrow BA$
$A \rightarrow a$
$B \rightarrow b$

G3: $S \rightarrow ABCS$ | e tutte le permutazioni della stringa ABCS $\mid \varepsilon$
$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow c$

G4: $S \rightarrow ABCS \mid \varepsilon$
$BA \rightarrow AB$
$AB \rightarrow BA$
$CB \rightarrow BC$
$BC \rightarrow CB$
$AC \rightarrow CA$
$CA \rightarrow AC$
$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow c$

Si dica, motivando brevemente la risposta, se G1 e G2, e G3 e G4 sono tra loro equivalenti.

Parte B
Si dica, motivando brevemente la risposta, se i linguaggi generati da G1, G2, G3 e G4 sono riconoscibili da automi a pila (anche nondeterministici).

(continua sul retro)

Esercizio 2 (Punti 12)

Per una funzione $F:N \rightarrow N$, un elemento x del suo dominio di definizione è detto *mimimo globale di F* se F(x) è definita e, per ogni valore y in cui F(y) è definita, $F(x) \leq F(y)$, dove $\leq$ definisce l'usuale relazione di "minore o uguale" tra numeri naturali.

1. Si determini se il problema di trovare se una generica funzione computabile f ha un minimo globale è decidibile.
2. Si determini se lo stesso problema è semidecidibile.
3. Come cambia, se cambia, la risposta al quesito precedente se si considerano solo funzioni totali e computabili?

Esercizio 3 (Punti 10)

Si consideri il problema della ricerca di un elemento x in un array S di lunghezza n. In tutto l'esercizio, consideriamo solo la complessità temporale e il criterio di costo costante.

Come noto, l'algoritmo più immediato per risolvere il problema è quello della ricerca sequenziale, che scandisce tutto l'array sequenzialmente alla ricerca dell'elemento x, e raggiunge una complessità di $\Theta(n)$ nel caso pessimo.

Si consideri ora il problema di trovare un algoritmo per risolvere lo stesso problema che sia *asintoticamente peggiore* della ricerca sequenziale, ossia raggiunga una qualche complessità $\Theta(p(n))$, con p(n) funzione totale e computabile, tale che $\lim_{n \to \infty} \frac{n}{p(n)} = 0$.

1. Si descriva un algoritmo di ricerca *asintoticamente peggiore* della ricerca sequenziale, e se ne valuti la complessità temporale, mostrando che è effettivamente asintoticamente peggiore.
2. Un algoritmo A che risolve un problema P con complessità $T_A(n)$ è detto *pessimo* se ogni altro algoritmo B che risolve P con complessità $T_B(n)$ è tale che: o A è asintoticamente peggiore di B, oppure $T_A(n)$ $\Theta(T_B(n))$.
   Per un generico problema P, esiste sempre un algoritmo pessimo che lo risolve? Esiste un problema P che ammette un algoritmo pessimo per la sua soluzione? Motivare brevemente le risposte.

Soluzioni

Esercizio 1

Parte A

G1 e G2 sono equivalenti. Infatti entrambe generano tutte e sole le stringhe con ugual numero di a e b.

G3 e G4 non lo sono. Infatti G3, pur generando stringhe con ugual numero di a, b, c, genera solo sequenze di triple di simboli, costituite ognuna da tre simboli (a, b, c) in un ordine qualsiasi, quindi non può generare, ad esempio, stringhe del tipo $a^n b^n c^n$, che possono invece essere generate da G4. Da ciò si intuisce anche che per riconoscere le stringhe di L(G3) non sia necessario effettuare un conteggio "globale" dei caratteri a, b, c.

Parte B

Poiché G1 e G3 sono non contestuali, i linguaggi da esse generati sono certamente riconoscibili da automi a pila. Essendo G2 equivalente a G1, lo stesso vale per L(G2).

Invece G4 richiede il riconoscimento di stringhe del tipo $a^n b^n c^n$ che devono essere distinte, ad esempio, da stringhe come $a^n b^n c^{2n}$. Ciò richiede una capacità di "conteggio illimitato doppio" che, come ben noto, non rientra tra le possibilità dell'automa a pila.

Esercizio 2

1. Il problema non è decidibile, in quanti riducibile al problema di determinare se una generica funzione computabile è definita per almeno un valore del suo dominio, problema che non è decidibile.
   Infatti, sulla base della definizione data, tenendo conto che l'insieme dei numeri naturali è discreto e limitato inferiormente, una funzione computabile $f: N \rightarrow N$ ha minimo globale se e solo se è definita in almeno un punto.
2. Lo stesso problema è semidecidibile, perchè lo è la proprietà di essere definita in almeno un punto (usando l'usuale costruzione diagonale).
3. Sia f una funzione totale e computabile, e sia $I \subseteq N$ la sua immagine. Essendo I un insieme limitato inferiormente (da 0), è sicuramente definito l'estremo inferiore i = inf I. Inoltre, essendo I un insieme discreto, i appartiene a I, quindi i = min I. Il valore x tale che f(x) = i è il minimo di f, che dunque esiste sempre per funzioni totali e computabili a valori in N. Dunque il problema è deciso, e quindi a maggior ragione decidibile.
   Per ripassare un pò di matematica, ecco un'altra giustificazione che I ha sicuramente minimo. Un insieme S è detto ben ordinato (well-ordered) sse ogni suo sottinisieme non vuoto ha un elemento minimo. Per assurdo, sia I *non* ben ordinato. Allora esiste un suo sottoinsieme $\emptyset \neq J \subseteq I$ che non ammette minimo. Sia K il complemento, rispetto a N, di J. Se $0 \in J$, esso è anche minimo; quindi $0 \notin J$ e dunque $0 \in K$. Per induzione, tutti i naturali appartengono a K e dunque J è vuoto, contrariamente all'ipotesi. Dunque I è ben ordinato, e dunque ha minimo.

Esercizio 3

1.

Data una qualunque funzione computabile p(n) tale che $\lim_{n\to\infty} \frac{n}{p(n)} = 0$, si può generare un algoritmo di ricerca che abbia complessità Ω(p(n)) semplicemente "peggiorando" un normale algoritmo di ricerca sequenziale aggiungendo l'esecuzione di un numero di istruzioni pari a p(n), ove n è la lunghezza dell'array ove si sta cercando. In pseudocodice:

```
ricerca_p_peggiore(x, S):
1       n = |S|
2       max = p(n)
3       for i:=1 to max do
4               continue;
5       return ricerca_sequenziale(x, S)
end
```

La complessità dell'istruzione 5 è chiaramente pari a Θ(n). Sia c(n) la complessità dell'istruzione 2, ossia il costo di calcolare p(n). Infine il ciclo **for** delle istruzioni 3-4 ha complessità Θ(p(n)). In totale quindi, ricordando che p(n) cresce asintoticamente più di n, si ha una complessità Θ(c(n) + p(n)) che è Ω(p(n)).

2.

Nella soluzione di 1. abbiamo mostrato come la complessità si possa "peggiorare" arbitrariamente. Dunque per qualsiasi problema non ha senso cercare la complessità pessima perchè è sempre possibile scegliere una funzione che cresce più velocemente e peggiorare l'algoritmo di quel fattore.