Algoritmi e Principi dell'Informatica

Appello del 6 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 9)

Punto a)

Si scriva una grammatica che generi il linguaggio L1 composto da stringhe sull'alfabeto $\Sigma = \{(,),]\}$, tali che le parentesi tonde siano bilanciate — ossia ben parentetizzate —, con la variante che la parentesi quadra può bilanciare un numero arbitrario (non nullo) di parentesi tonde aperte. Ad esempio, le stringhe (()()), ((())), (((()[(()] appartengono al linguaggio, mentre le stringhe ((), ((((](,]()() non vi appartengono. Si scriva una grammatica, preferibilmente a potenza minima che produca il linguaggio L1.

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.

Punto b)

E' vero che se un linguaggio L non è regolare allora neanche il suo complemento L^c è regolare? Perché?

Punto c)

E' vero che se un linguaggio L non è libero dal contesto allora neanche il suo complemento L^c è libero dal contesto? Perché?

Esercizio 2 (punti 7)

Si consideri l'insieme S delle stringhe costituite da caratteri appartenenti a due alfabeti disgiunti, A1 e A2. Si formalizzi, mediante una formula del prim'ordine, il predicato unario `proprietà`, che indica che il suo argomento appartiene al sottoinsieme di S costituito da stringhe in cui non compaiano due caratteri consecutivi appartenenti ad A1. Ad esempio, se i caratteri di A1 sono lettere maiuscole e quelli di A2 lettere minuscole, le seguenti stringhe soddisfano la proprietà suddetta:

AbcDghm, cvHmUnI, ε, F, …

mentre le seguenti non la soddisfano:

AA, aGnnHH, …

NB: La formula deve far uso esclusivamente dei simboli seguenti:

- Quantificatori e connettivi logici proposizionali
- Il simbolo di appartenenza ∈
- Simboli di variabile i cui domini siano insiemi di stringhe o insiemi di caratteri
- I simboli di costante A1 e A2 rappresentanti i due alfabeti
- Il simbolo '•' che denota l'operazione di concatenazione, applicabile a stringhe e a caratteri.

Esercizio 3 (punti 6)

Si immagini di voler ordinare un array di n elementi mediante l'algoritmo Quicksort. Si immagini inoltre che l'array in input sia tale per cui l'operazione di partizione sposti il pivot in posizione n – k, k essendo una costante indipendente da n, e si assuma che lo stesso effetto si ripresenti nella successiva partizione della porzione di array di lunghezza n – k -1; e così via finché la cardinalità della porzione considerata non diventa <= k.

Sulla base delle suddette informazioni, è possibile determinare l'ordine di grandezza della complessità temporale dell'esecuzione di Quicksort su un tale array? In caso positivo, quale sarebbe l'ordine di grandezza?

Esercizio 4 (punti 11)

Sia L una lista semplice (cioè con puntatori solo all'elemento successivo) contenente *n* interi positivi memorizzati in ordine *strettamente crescente*. Si consideri il problema di trovare due puntatori a due elementi di L, *i*, e *j*, *se* esistono, tali che la somma degli elementi di L compresi tra *i* e *j*, *estremi inclusi*, sia uguale a un valor dato X.

Si descriva un algoritmo, mediante opportuno pseudocodice, per risolvere il problema (ossia stabilire se tali *i, j* esistono e nel caso produrli in uscita) e se ne valuti la complessità asintotica.

NB: la complessità dell'algoritmo, ovviamente, influenza la valutazione della qualità dell'algoritmo prodotto.

Soluzioni schematiche

Esercizio 1

Punto a)

Una grammatica noncontestuale è necessaria e sufficiente per generare L1; ad esempio la seguente:

$S \rightarrow BS \mid N]S \mid \varepsilon$

$B \rightarrow BB \mid (B) \mid \varepsilon$

$N \rightarrow NN \mid (N \mid (B$

Punto b)

Vero. Supponiamo per assurdo che L^c sia regolare. Allora grazie alla chiusura dei linguaggi regolari rispetto alla complementazione, anche il complemento di L^c dev'essere regolare. Ma il complemento di L^c è L, che per ipotesi non è regolare. Assurdo.

Punto c)

Falso. Un esempio di linguaggio non libero dal contesto è $L=\{a^n b^n c^n\}$, non riconoscibile da nessun (ND)PDA. Il suo complemento è invece libero dal contesto. Si tratta infatti del linguaggio le cui stringhe o sono del tipo $a^n b^m c^l$ con $n \neq m$ o $n \neq l$ oppure non sono del tipo a*b*c*. Chiamiamo $L_1=\{ a^n b^m c^l \mid n \neq m \text{ o } n \neq l \}$ e L_2=a*b*c* e sia L_2^c il complemento di L_2. In sostanza, L^c è dato dall'unione di L_1 e L_2^c, che sono entrambi liberi dal contesto. Infatti, per riconoscere L_1 basta un NDPDA, mentre notiamo che L_2 è un linguaggio regolare, e quindi anche il suo complemento L_2^c è regolare (e quindi anche libero dal contesto).

Esercizio 2

$\forall x \text{ proprietà}(x) \leftrightarrow \neg\exists y, z, c_1, c_2(c_1 \in A1 \wedge c_2 \in A1 \wedge x = y \bullet c_1 \bullet c_2 \bullet z)$

Esercizio 3

L'equazione alle differenze in questo caso è la seguente

$T(n) = T(k) + T(n-k) + \Theta(n)$

laddove $T(k) = O(1)$, in quanto per ordinare un array di dimensione fissata *k* serve un tempo costante, quindi la soluzione $O(n^2)$

Esercizio 4

```
FIND_SUM(L,X)
1  j := L.head
2  i := L.head
3  sum := j.key
4  while i ≠ j.next and j ≠ NIL and sum ≠ X
5    if sum < X
6      j := j.next
7      sum := sum + j.key
8    else
9      sum := sum – i.key
10     i := i.next
11 if sum = X
12   return (i, j)
13 else
14   return false
```

La complessità asintotica dell'algoritmo è chiaramente O(n) poiché *i* e *j* scorrono la lista una sola volta.