Informatica Teorica

Esame del 16 Luglio 2009 **Tempo a disposizione: 2h**

Esercizio 1 (12 punti)

Si consideri il linguaggio

$L1 = \{ x \in \{a,b,c,d\}^* \mid \#a(x) - \#b(x) = \#c(x) - \#d(x) \}$

cioè l'insieme delle stringhe di alfabeto {a,b,c,d} il cui la differenza tra il numero delle a e delle b sia uguale alla differenza tra il numero delle c e delle d.

Si consideri ora il linguaggio

$$L2 = \{ x \in \{a,b,c,d\}^* \mid \exists k(|x| = 2k) \wedge \forall y \in \{a,b,c,d\}^* (\exists z \in \{a,b,c,d\}^* (x = y.z) \wedge \exists k(|y| = 2k) \rightarrow \#a(y) - \#b(y) = \#c(y) - \#d(y)) \}$$

1. Descrivere a parole il linguaggio L2. In che relazione stanno i due linguaggi L1 ed L2?
2. Si scriva un automa riconoscitore a potenza minima che riconosce il linguaggio L1.
3. Si scriva un automa riconoscitore a potenza minima che riconosce il linguaggio L2.

Esercizio 2 (7 punti)

Si consideri l'insieme delle funzioni computabili, aventi come argomento e come valore un numero naturale, che sono definite solo per valori dispari dell'argomento (o, detto diversamente, che non sono definite per alcun argomento pari: si noti che tale insieme include anche la funzione ovunque indefinita).

1. Adottando l'usuale notazione impiegata nel corso, scrivere una formula logica del primo ordine, con una variabile libera *x*, che funga da predicato caratteristico di tale insieme, cioè la formula sia vera esattamente per i valori di *x* che corrispondono agli elementi dell'insieme.
2. Dire se l'insieme in questione è decidibile.
3. Dire se l'insieme è semidecidibile.

Esercizio 3 (14 punti)

1. Descrivere a parole (ma in modo sufficientemente preciso per poterne valutare la complessità) una MT a nastro *singolo* che, data in input una sequenza di caratteri alfabetici minuscoli, trova il simbolo che è a metà esatta della sequenza (si supponga pure che la sequenza abbia un numero dispari di simboli).
Dare la complessità (sia temporale che spaziale) della MT così ideata.

2. Descrivere a parole (ma in modo sufficientemente preciso per poterne valutare la complessità) una MT a nastro *singolo* che, data in input una sequenza di caratteri alfabetici minuscoli, la ordina in ordine lessicografico crescente.
Quale è la complessità (sia temporale che spaziale) della MT descritta?

3. Descrivere a parole (ma in modo sufficientemente preciso per poterne valutare la complessità) una MT a nastro *singolo* che, data in input una sequenza di numeri naturali codificati in binario (separati uno dall'altro dal simbolo #), la ordina in ordine crescente.
Quale è la complessità (sia temporale che spaziale) della MT descritta?

NB1: Come "parametri di dimensione dati" al punto 3 si usino il numero *n* di naturali presenti nella sequenza e il valore massimo *k* presente nella sequenza. Si supponga pure che ogni valore nella sequenza sia codificato con lo stesso numero di cifre binarie (corrispondente al numero di cifre necessario per codificare *k*).

NB2: Per tutti e tre i punti dell'esercizio, il punteggio massimo verrà assegnato se la MT ideata non usa altre celle del nastro oltre a quelle contenenti la stringa di input.

Soluzioni

Esercizio 1

1. L2 è l'insieme delle stringhe di lunghezza pari, nelle quali tutte le stringhe prefisso di lunghezza pari la differenza tra il numero delle a e delle b è uguale alla differenza tra il numero delle c e delle d. Quindi L2 è un sottoinsieme di L1
2. Se #a(x)-#b(x)=#c(x)-#d(x) allora #a(x)+#d(x)=#b(x)+#c(x), quindi durante la scansione della stringa occorre contare la differenza tra la somma delle a e delle d e la somma delle b e delle c. Tale differenza può crescere senza limiti con la lunghezza della stringa, quindi è necessario, e sufficiente, un automa a pila che memorizzi in forma unaria, sulla pila il valore della differenza.
3. Nelle stringhe di L2 la differenza tra la somma delle a e delle d e la somma delle b e delle c in un qualsiasi prefisso deve avere valore assoluto <1, quindi è sufficiente un automa a stati finiti.

Esercizio 2

1. $\forall z(f_x(z)\neq\perp\rightarrow\exists k(z=2k))$
2. No, per il teorema di Rice.
3. No, perché è semidecidibile il complemento dell'insieme in questione, cioè quello delle funzioni che sono definite per qualche valore *pari* dell'argomento, cosa che si può dimostrare con l'usuale argomento "diagonale": si eseguono a turno le funzioni per un numero crescente di passi, sugli argomenti pari.

Esercizio 3

1.

Il modo più efficiente per trovare il punto di mezzo di una sequenza mediante una MT a nastro singolo si basa sull'idea di "marcare" la prima e l'ultima lettera della sequenza (per esempio cambiando a in a', oppure b in b', ecc.), poi scorrere la sequenza avanti ed indietro, spostando i "marker" via via un simbolo avanti ed un simbolo indietro, fino a che essi non si sovrappongono.
La complessità temporale di una simile MT è $\Theta(n^2)$, mentre la complessità spaziale è $\Theta(n)$ (in effetti è *esattamente* n, che è la minima complessità spaziale possibile per MT a nastro singolo, in quanto il nastro di memoria contiene almeno la stringa di input).

2.

Una maniera efficiente di ordinare la sequenza desiderata è mediante un algoritmo di bubblesort, in cui elementi contigui e disordinati della sequenza (tali cioè il simbolo a sinistra è maggiore di quello a destra) sono scambiati tra loro fino a che la sequenza è ordinata. Più precisamente, la stringa viene scorsa da sinistra a destra, ed ogni volta che si trova un simbolo che è maggiore del simbolo immediatamente successivo, essi vengono scambiati. Inoltre, se viene fatto uno scambio di simboli, la MT tiene traccia nei suoi stati del fatto di avere effettuato uno scambio. Quando arriva in fondo alla stringa, se uno scambio è stato effettuato, la MT torna all'inizio del nastro e ricomincia a scorrerlo. La computazione termina quando, arrivando in fondo al nastro, non sono stati effettuati scambi di simboli.
La complessità temporale di una tale MT è $\Theta(n^2)$, in quanto il meccanismo è lo stesso del bubblesort classico studiato in corsi di algoritmi, mentre la complessità spaziale è $\Theta(n)$. In effetti, in questo caso la complessità spaziale è *esattamente* n, in quanto non serve memorizzare nulla sul nastro della MT a parte la stringa in ingresso.
Si noti come l'algoritmo di COUNTING-SORT che, nel caso di esecuzione su MT a k nastri o nel caso di macchina RAM sarebbe più efficiente (lineare per MT a k nastri, $\Theta(n \log(n))$ per macchina RAM a costo logaritmico), nel caso di MT a nastro singolo avrebbe comunque complessità quadratica (ed in compenso richiederebbe celle aggiuntive oltre a quelle necessarie per memorizzare la stringa in ingresso).

3.
Anche in questo caso un algoritmo efficiente è quello di bubblesort, con lo stesso meccanismo descritto al punto 2. Tuttavia, in questo caso il costo di confrontare e scambiare numeri contigui non è costante, ma è, $\log(k)^2$, in quanto richiede, alla peggio, di spostare di log(k) elementi la testina avanti e indietro per log(k) volte.
Quindi, la complessità temporale dell'algoritmo è $\Theta(n^2 \log(k)^2)$. La complessità spaziale è invece $\Theta(n \log(k))$, in quanto non è necessario usare altra memoria oltre a quella che serve per memorizzare la sequenza di dati di input (si noti che $\Theta(n \log(k))$ è in effetti la lunghezza della stringa in input alla MT).