Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

10 febbraio 2023

Informatica teorica

Esercizio 1 (8 punti)
Si consideri il linguaggio $L = \{0^n \mid n \in \mathbb{N} \text{ e } 0^n \text{ appare nell'espansione decimale di } \pi\}$. Il linguaggio L è regolare? Si motivi esaurientemente la risposta.

Soluzione

Si danno solamente due casi:

1. esiste una sequenza "massima" 0^k che appare nell'espansione decimale di π tale per cui la sequenza 0^m, con $m > k$, non appare nell'espansione decimale di π;

2. qualunque sequenza di 0, di qualunque lunghezza, appare nell'espansione decimale di π.

In entrambi i casi il linguaggio è regolare, in quanto esprimibile mediante un'espressione regolare. Nel primo caso, l'espressione regolare è $\varepsilon|0|00|\ldots|0^k$; nel secondo, 0^*.

Esercizio 2 (8 punti).

Dato un linguaggio L definito su un alfabeto I, si consideri l'ipotetica macchina di Turing M_L a k nastri, che ignora la stringa in input e stampa sul nastro di output la sequenza $l_0 \diamond l_1 \diamond l_2 \diamond \ldots$, dove $l_i \in L, i \in \mathbb{N}$ e $\diamond$ è un simbolo non presente in I. Le parole di L compaiono una ed una sola volta nell'output di M_L, in un ordine non noto a priori.

1. Si dimostri che, se M_L esiste, allora L è semidecidibile.

2. Si consideri il caso in cui M_L esiste e le parole $l_i \in L$ compaiono in ordine lessicografico nel nastro di output di M_L. Si indichi se L è decidibile, semidecidibile o indecidibile e lo si dimostri.

Chi ha diritto alla riduzione del 30% della prova, svolga unicamente il primo dei due punti.

Soluzione

1. L'esistenza di M_L consente di ricavare direttamente un semi-algoritmo (ovvero una procedura algoritmica di decisione che termina unicamente nel caso in cui la risposta sia 1) che calcola la funzione caratteristica di L (ovvero la funzione che vale 1 se $x \in L$, 0 altrimenti), $c_L(x)$, se x appartiene a L: è sufficiente emulare M_L ispezionando il nastro di output e, nel momento in cui x compare su di esso, restituire 1. Dato che tutte le parole di L compaiono sul nastro di output, in un tempo arbitrariamente grande, ma finito, comparirà anche x.

2. L è almeno semidecidibile per quanto già mostrato al punto precedente. Si può dimostrare che L è decidibile costruendo un algoritmo a partire da M_L che calcola $c_L(x)$: è sufficiente emulare M_L fino a quando si incontra x o una parola y che segue x in ordine lessicografico. Nel primo caso si restituisce 1 (x appartiene a L), nel secondo caso 0 (x non appartiene ad L in quanto x sarebbe dovuta comparire prima di y nell'output di M_L).

Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

10 febbraio 2023

Algoritmi e strutture dati

Esercizio 3 (8 punti)
Si consideri il seguente linguaggio L definito sull'alfabeto I delle parentesi tonde e quadre. Una parola $l \in L$ è ottenuta partendo da una sequenza ben parentetizzata di sole parentesi tonde t e da una sequenza ben parentetizzata di sole parentesi quadre q concatenando in ordine un simbolo di t e uno di q. A titolo di esempio, partendo da $t = ()(())$ e $q = [][[]]$, otteniamo $l = ([)]([(])])]$.

Si descrivano un algoritmo per macchina di Turing a $k = 1$ nastri ed uno per macchina RAM che, data una generica stringa di parentesi tonde e quadre, restituiscono 1 se essa appartiene a L, 0 altrimenti. Si privilegino soluzioni a complessità temporale minima. Si descrivano le complessità spaziali e temporali di entrambi, con tutti i criteri di costo applicabili.

Chi ha diritto alla riduzione del 30% della prova, descriva ed analizzi unicamente l'algoritmo per macchina di Turing a $k = 1$ nastri.

Soluzione

Un schema di algoritmo per macchina di Turing a $k = 1$ nastri è il seguente:

1. Il nastro di lavoro della MT conterrà due contatori binari che tengono traccia delle parentesi (tonde e quadre) aperte non ancora accoppiate ad una chiusa. I due contatori sono separati da un simbolo c diverso da 0 e 1. Il simbolo c viene quindi posizionato sul nastro di lavoro come prima mossa.

2. Per ogni simbolo letto dal nastro di input:

   - Se si tratta di una parentesi aperta, la testina del nastro di lavoro si sposta sul bit meno significativo del contatore relativo al tipo di parentesi letta (tonda o quadra) ed incrementa il contatore binario. Se necessario, sposta tutti i simboli del contatore (e di quello adiacente) per far posto ad un eventuale riporto.
   - Se si tratta di una parentesi chiusa, la testina del nastro di lavoro si sposta sul bit meno significativo del contatore opportuno e controlla se il contatore vale 0. In questo caso la macchina si arresta restituendo 0. Se il contatore ha un valore maggiore di zero, calcola un decremento del contatore. È possibile "ricompattare" i contatori nel caso di un prestito dalla cifra più significativa, ma non migliora la complessità asintotica del procedimento.

3. Controlla se sul nastro di lavoro sono rimasti due contatori con valore 0, nel qual caso restituisce 1, altrimenti restituisce 0.

La complessità temporale dell'algoritmo è $\mathcal{O}(n \log(n))$: si ha infatti che, per ogni parentesi letta, viene effettuato un incremento o decremento di un contatore binario (costo $\log_2(n)$) e, nel caso sia necessario, uno spostamento di due contatori binari (costo $\mathcal{O}(\log(n))$). La complessità spaziale dell'algoritmo è $\mathcal{O}(\log(n))$.

L'algoritmo per la macchina RAM è analogo, con l'unica differenza che i contatori binari sono salvati in due celle di memoria della macchina RAM. Si ha quindi che la complessità temporale dell'algoritmo per la macchina RAM a criterio di costo costante è $\mathcal{O}(n)$, mentre quella spaziale è $\mathcal{O}(1)$, dato che vengono impiegate solo due celle di memoria. La complessità temporale e spaziale dell'algoritmo per macchina RAM coincidono con quelle della MT a k nastri nel caso sia utilizzato il criterio di costo logaritmico.

Esercizio 4 (8 punti)

Si consideri una struttura dati Q che possa funzionare sia da coda che da pila, usando una lista semplice, cioè dove ogni nodo possiede un unico puntatore all'elemento successivo. Si implementino in pseudo-codice le seguenti operazioni e se ne valutino le complessità temporali:

1. *dequeue*(Q): rimuove da Q, restituendolo, il nodo di lista che è rimasto nella collezione Q più a lungo;
2. *pop*(Q): rimuove da Q, restituendolo, l'ultimo nodo inserito nella collezione Q;
3. *insert*(Q, x): inserisce un nodo x nella collezione Q;
4. *append*(Q, Q'): fonde le due collezioni Q e Q' in Q (l'effetto dev'essere lo stesso che si otterrebbe inserendo ciascun elemento di Q', dal più vecchio al più recente, in Q).

Soluzione

Q ha due attributi, *tail* e *head*, che puntano rispettivamente all'ultimo e al primo elemento della lista contenente i dati.

```
dequeue(Q):
    x := Q.head
    Q.head := x.next
    return x

pop(Q):
    x := Q.head
    y := x.next
    while y != Q.tail
        x := y
        y := y.next
    x.next := NIL
    Q.tail := x
    return y

insert(Q, x):
    Q.tail.next := x
    Q.tail := x

append(Q, Q'):
    Q'.tail.next := Q.head
    Q.head := Q'.head
```

Le complessità temporali sono tutte costanti, a parte *pop* che è $\Theta(|Q|)$.