Algoritmi e Principi dell'Informatica

Soluzioni al Tema d'esame
07 Settembre 2020

1 Informatica teorica

Esercizio 1
Siano $A = \{0, 1, 2\}$ e $B = \{0, 1\}$. Si considerino i linguaggi $L_1 = \{x1x^R \mid x \in A^+\}$ e $L_2 = B^+212B^+$.

1. Si dica quali sono i modelli a *potenza minima* per L_1 ed L_2, motivando adeguatamente la risposta.
2. Si descriva il funzionamento di un automa (o si definisca una grammatica) a potenza minima per l'intersezione tra L1 e L2.

Soluzione

1. *Rispettivamente automa a pila non deterministico e automa a stati finiti.*
2. *Solito automa a pila* deterministico *che impila i caratteri in B, fino ad arrivare al primo 2; dopo il secondo 2 spila controllando la corrispondenza col carattere letto.*

Esercizio 2

Si dica, motivando opportunamente la risposta, se i problemi seguenti sono decidibili:

1. Stabilire se, dato un generico automa a stati finiti, esso riconosce tutte le stringhe in A^*, con $A = \{0, 1\}$;
2. Stabilire se, dato un generico automa a stati finiti, esso riconosce tutte e sole le stringhe che codificano numeri primi espressi in notazione binaria, con $A = \{0, 1\}$.

Come cambierebbe la risposta ai quesiti precedenti se fosse data una macchina di Turing al posto di un automa a stati finiti?
Soluzione

1. *Decidibile poiché è decidibile l'equivalenza tra FSA, quindi basta disegnarne uno che accetti tutto A^*.*
2. *Decidibile poiché la risposta è sempre no (nessun FSA è in grado di riconoscere il linguaggio dato).*

Le varianti con TM sono tutte indecidibili per il teorema di Rice.

2 Algoritmi e strutture dati

Esercizio 1
Si considerino i formalismi delle Macchine di Turing deterministiche a 1 nastro di memoria (MT-1) e degli Automi a 2 Pile deterministici (A2P). Un A2P è un automa a pila deterministico che ha a disposizione una pila aggiuntiva. Mostrare come una MT-1 può simulare il comportamento di un A2P e stabilire la complessità, nel caso pessimo, della MT-1 in funzione della complessità dell'A2P simulato, motivando opportunamente la risposta.
Soluzione

Si indichi con $T_{A2P}(n)$ la complessità di un generico A2P.

Per simulare un A2P con una MT-1 è sufficiente usare il nastro di memoria per simulare il contenuto delle due pile, per esempio mettendo prima il contenuto della prima pila, seguito da un simbolo separatore, seguito da contenuto della seconda pila. La simulazione della singola mossa dell'A2P richiede di scorrere il nastro di memoria fino a giungere alla cima della pila di sinistra, proseguire fino a giungere alla cima della pila di destra, e modificare opportunamente il nastro per simulare il cambiamento dei simboli in cima alla pila. A questo proposito, se le pile aumentano o diminuiscono di lunghezza, occorre scorrere il nastro di memoria, opportunamente spostando i simboli (per esempio, in caso di aggiunta di 2 simboli in cima alla pila di sinistra, il contenuto della pila di destra va tutto spostato di 2 posizioni a destra; viceversa, se il simbolo in cima alla pila di sinistra va cancellato, occorre spostare tutto il contenuto della pila di destra di una posizione a sinistra). Questo richiede, nel caso pessimo, complessità $O(T_{A2P}(n))$, quindi la complessità totale di una MT-1 che simula un A2P è $O((T_{A2P}(n))^2)$.

Esercizio 1
Si consideri il caso di una tabella di hash di 8 elementi, le cui chiavi sono costituite da numeri interi, gestita con la disciplina dell'indirizzamento aperto (open addressing). Per semplicità, si rappresenti lo stato di una tabella vuota nel seguente modo

[_, _, _, _, _, _, _, _]

considerando il bucket (slot) più a sinistra come quello di indice 0 e quello più a destra come quello di indice 7. Si indichi con $h(k, i)$ la funzione di hash $h : \mathbb{N} \times \mathbb{N} \to \{0, \ldots, 7\}$ dove k indica il valore della chiave e i l' indice del passo di ispezione (alla prima ispezione $i = 0$).

1. Si mostri lo stato della tabella, dopo ognuna delle seguenti operazioni

   - Inserimento di 2
   - Inserimento di 10
   - Inserimento di 18
   - Cancellazione di 10
   - Inserimento di 26

   considerando $h(k, i) = (k + i) \bmod 8$

2. Si consideri l'uso della seguente funzione di hash $h(k, i) = (k + h_2(k)i) \bmod 8$, dove la funzione $h_2(k)$: *a)* codifica k in binario naturale, *b)* toglie tutti gli zeri consecutivi presenti nelle posizioni meno significative della codifica binaria di k, *c)* interpreta il risultato come la codifica in binario naturale di un intero. La funzione indicata rappresenta una buona scelta per la tabella di cui sopra, ovvero consente alla sequenza di ispezione di visitare tutti i bucket?

Soluzione

1. *Stato della tabella:*
   - *Inserimento di 2:* `[_,_,2,_,_,_,_,_]`
   - *Inserimento di 10:* `[_,_,2,10,_,_,_,_]`
   - *Inserimento di 18:* `[_,_,2,10,18,_,_,_]`
   - *Cancellazione di 10:* `[_,_,2,`$\times$`,18,_,_,_]`
   - *Inserimento di 26:* `[_,_,2,26,18,_,_,_]`

2. *La funzione $h_2(k)$ ha la proprietà di garantire un risultato dispari (a tutti gli effetti, rimuove il fattore $2^i, i \in \mathbb{N}$ dalla fattorizzazione di k), tranne nel caso in cui il valore dato in ingresso sia zero. Si ha di conseguenza che la strategia di double hashing indicata avrà una sequenza di ispezione che tocca tutte le celle della tabella di hash, in quanto il risultato di $h_2(k)$ è coprimo con la lunghezza della tabella, salvo appunto nel caso di $h(0, i)$ per qualunque valore di i. La funzione di hash non è quindi una buona funzione di hash, considerando il criterio del raggiungimento di tutti i bucket (slot) durante le ispezioni.*