Algoritmi e Principi dell'Informatica

Tema d'esame del 29 Giugno 2021

1 Informatica teorica

Esercizio 1 (8 punti)

a) Scrivere una grammatica a potenza minima che generi il linguaggio seguente:

$L_1 = \{(bc)^m \mid m \geq 2\}$

b) Sfruttando la risposta al punto precedente, scrivere una grammatica a potenza minima che generi il linguaggio seguente:

$L_2 = \{b^k a^\ell (bc)^m a^n b^p \mid k > p, 2\ell = n, m \geq 2\}$

Soluzione

a) È sufficiente una grammatica regolare, dove S_0 è l'assioma.

$$\begin{array}{lcl} S_0 & \rightarrow & bS_1 \\ S_1 & \rightarrow & cS_2 \\ S_2 & \rightarrow & bS_3 \\ S_3 & \rightarrow & c \mid cS_2 \end{array}$$

b) È sufficiente una grammatica non contestuale. Il linguaggio può infatti anche essere pensato nel modo seguente:

$L_2 = \{b^s b^p a^\ell (bc)^m (aa)^\ell b^p \mid s \geq 1, p \geq 0, \ell \geq 0, m \geq 2\}$

$$\begin{array}{lcll} S & \rightarrow & UV & \\ U & \rightarrow & bU \mid b & \text{1 o più } b \\ V & \rightarrow & bVb \mid W & \text{generiamo } p \text{ lettere } b \text{ da ambo i lati} \\ W & \rightarrow & aWaa \mid S_0 & \text{generiamo } \ell \text{ lettere } a \text{ a sx e } 2\ell \text{ a dx} \end{array}$$

dove S_0 è il simbolo iniziale della grammatica per L_1.

Esercizio 2 (8 punti)

Sia L_1 un linguaggio ricorsivamente enumerabile, L_2 un linguaggio regolare e $L = L_1 \cap L_2$ la loro intersezione. Per ciascuna delle seguenti classi di linguaggi, si argomenti se L può farne parte, deve necessariamente farne parte o non può farne parte:

a) linguaggi regolari;

b) linguaggi ricorsivi;

c) linguaggi ricorsivamente enumerabili.

Motivare opportunamente e fornire esempi laddove appropriato.

Soluzione

a) Può farne parte: se L_1 è regolare allora anche L è regolare (chiusura rispetto a intersezione); ad esempio, se $L_1 = \emptyset$ abbiamo $L = \emptyset$. Può non farne parte: se L_1 non è regolare e $L_2 = A^*$, dove A è l'alfabeto, allora L non è regolare.

b) In modo simile al punto precedente, se L_1 è ricorsivo allora anche L è ricorsivo (chiusura rispetto a intersezione), altrimenti potrebbe non esserlo. La ragione è che L_2, essendo regolare, è anche ricorsivo.

c) L è certamente ricorsivamente enumerabile in quanto intersezione di due linguaggi ricorsivamente enumerabili (chiusura rispetto a intersezione), dato che L_2, essendo regolare, è anche ricorsivamente enumerabile.

2 Algoritmi e strutture dati

Esercizio 3 (8 punti)
Si consideri un albero binario di ricerca T contenente n nodi. Si implementino in pseudocodice le seguenti procedure, e se ne analizzi la complessità temporale.

a) `simmetrico(T)`: La procedura controlla se l'albero ha struttura simmetrica rispetto all'asse verticale passante per la radice.

b) `gemelli(T)`: La procedura controlla se i sottoalberi figli della radice hanno la stessa struttura, ovvero se i sottoalberi sono sovrapponibili con una traslazione orizzontale rigida.

c) `stampa_strati(T)`: La procedura stampa il contenuto dell'albero per livelli, a partire dalla radice, stampando i nodi di ogni livello da sinistra a destra.

Soluzione

a)
```
simmetrico(T){
  return visitas(T.left,T.right)
}
visitas(L,R){
   if ((L == NIL) and (R == NIL))
      return true
   if ((L == NIL) or (R == NIL))
      return false
   return visitas(L.left,R.right) and visitas(L.right,R.left)
}
```

Complessità temporale $\Theta(n)$.

b)
```
gemelli(T){
  return visitag(T.left,T.right)
}
visitag(L,R){
   if ((L == NIL) and (R == NIL))
      return true
```

```
    if ((L == NIL) or (R == NIL))
       return false
    return visitag(L.left,R.left) and visitag(L.right,R.right)
}
```

Complessità temporale $\Theta(n)$.

c)
```
stampa_strati(T){
  if (not (T == NIL))
     Q.enqueue(T)
  while(not Q.is_empty())
     E = Q.dequeue()
     print(E)
     if (not (E.left == NIL))
        Q.enqueue(E.left)
     if (not (E.right == NIL))
        Q.enqueue(E.right)
}
```

Complessità temporale $\Theta(n)$.

Esercizio 4 (8 punti)
Si consideri una tabella hash di 1024 elementi, organizzata con indirizzamento aperto. La funzione di hash utilizzata è $h(k) = 4k \bmod 1024$, la sequenza di ispezione è $h'(k, i) = (4k + 2i) \bmod 1024$.

a) Qual è il numero massimo di elementi inseribili nella tabella?

b) Qual è la complessità attesa di accesso ad un elemento dopo l'inserimento di 256 elementi? (si assuma che la sequenza di ispezione sia tale da rispettare l'ipotesi di hashing uniforme rispetto ai bucket che possono essere raggiunti da essa).

Soluzione

a) 512; la funzione di hash emette unicamente chiavi multiple di 4 e la sequenza di ispezione procede a passo 2. Tutti i bucket della tabella con indice dispari saranno sicuramente inutilizzati.

b) Data la funzione di hash e sequenza di ispezione, la tabella è equivalente ad una con 512 bucket. Abbiamo quindi che il numero di tentativi medi prima di accedere ad un elemento è $(\frac{256}{512})^{-1} \log(\frac{1}{1-\frac{256}{512}}) \approx 1,38$.