Algoritmi e Principi dell'Informatica

Appello del 30 Giugno 2016

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 8)

Scriva una formula logica del prim'ordine che formalizzi la frase seguente:

Tutte le volte che mangio del salame entro 24 ore mi viene mal di pancia, a meno che, prima che sopraggiunga il mal di pancia, non prenda una pillola gastroprotettrice.

Si supponga, per semplicità, che mangiare salame sia un evento isolato nel tempo e che non possa ripetersi più di una volta nell'arco di 24 ore.

Suggerimento (non imposizione!)

Si consiglia di far uso, non necessariemtne esclusivo, dei seguenti predicati, tutti parametrici rispetto alla variabile t:

- Salame(t): mangio del salame al tempo t
- MalPancia(t): mi viene mal di pancia
- Pillola(t): assumo una pillola

La variabile t può essere interpretata indifferentemente in un dominio discreto o continuo.

Esercizio 2 (Punti 7)

Siano L_1, L_2, .. L_k, $k > 1$ linguaggi definiti su un alfabeto Σ, tali che valgano le seguenti proprietà:

- $\forall i \neq j, L_i \cap L_j = \emptyset$,
- $L_1 \cup L_2 \cup \ldots \cup L_k = \Sigma^*$,
- $\forall\ i, L_i$ è semidecidibile.

Si dica, giustificando brevemente la risposta se le seguenti affermazioni sono vere o false:

- Tutti i linguaggi L_1, L_2, .. L_k sono ricorsivi
- E' possibile che alcuni di essi siano regolari
- Tutti i linguaggi L_1, L_2, .. L_k sono necessariamente regolari

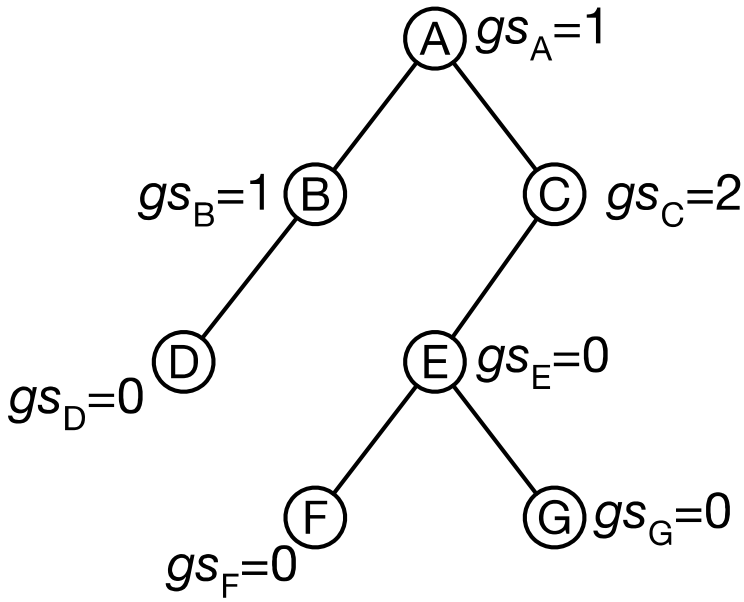
Esercizio 3 (Punti 7)

Si definisca una MT a k nastri che riconosca il linguaggio L = {www | w $\in$ $\{0,1\}^+$} e se ne valutino le complessità spaziale e temporale.

Esercizio 4 (Punti 8)

In un albero binario, il ***grado di sbilanciamento di un nodo*** può essere calcolato come valore assoluto della differenza fra il numero di foglie presenti nei suoi due sottoalberi. A partire da tale valore calcolato per ogni nodo, è possibile ricavare un ***grado di sbilanciamento di un albero*** come **il massimo tra i gradi di sbilanciamento** calcolati per tutti i suoi nodi.

Per esempio, dato il seguente albero binario, per il quale si riportano anche i gradi di sbilanciamento (*gs*) dei singoli nodi:



il grado di sbilanciamento per l'albero radicato in A è *2*, poiché questa è la differenza massima del numero di foglie nei due sottoalberi di A (quella cioè calcolata per il sottoalbero radicato in C).

Si definisca un algoritmo per il calcolo del grado di sbilanciamento di un albero binario; se ne discuta quindi la complessità.

Tracce di soluzioni

Esercizio 1

Soluzione A

I seguenti predicati

salame(t), malPancia(t), pillola(t) sono usati con l'ovvio significato. La frase dell'esercizio è allora formalizzata come segue:

$\forall$ t (salame(t) $\rightarrow$

$\forall$ t_1(t < t_1 <= t +24 $\rightarrow \neg$ salame(t_1)) $\wedge$

(

($\exists$ t_1 ((pillola(t_1) $\wedge$ ($\forall$ t_2 (t < t_2 < t_1 $\rightarrow$ $\neg$ malPancia(t_2))) $\rightarrow$

$\forall$ t_1(t <= t_1 <= t +24 $\rightarrow \neg$ malPancia (t))

$\vee$

($\exists$ t_3 (t <= t_3 <= t +24 $\wedge$ (($\forall$ t_2 (t < t_2 < t_3 $\rightarrow \neg$pillola(t_2)) $\rightarrow$ malPancia(t_3))

)

)

Soluzione B

$\forall$t (salame(t) $\rightarrow$

$\forall$ t_1(t < t_1 <= t +24 $\rightarrow \neg$ salame(t_1)) $\wedge$

(

$\exists t_1$ (t $\leq$ t_1 $\leq$ t +24 $\wedge$ pillola(t_1) $\wedge \forall t_2$ (t $\leq$ t_2 $\leq$ t+24 $\rightarrow \neg$malPancia(t_2)))

$\vee$

$\exists t_1$ (t $\leq$ t_1 $\leq$ t +24 $\wedge$ malPancia(t_1) $\wedge \forall t_2$ (t $\leq$ t_2 $\leq$ t_1 $\rightarrow \neg$pillola(t_2)))

)

)

Esercizio 2

- Poiché i linguaggi sono tra loro disgiunti e la loro unione è l'intero Σ^*, enumerando tutte stringhe di L_1, L_2, .. L_k nell'ordine 1, 2, …k, (ossia, una stringa di L_1, una stringa di L_2, …) prima o poi una qualsiasi string x deve comparire in una e una sola delle sequenze L_i, e quindi si può decidere a quale linguaggio appartiene.
- E' certamente possibile che qualche o anche tutti gli L_i, siano regolari, ma non necessario.

Esercizio 3

1) La MT fa una prima passata sulla stringa in ingresso (sia essa x) e sul nastro 1 memorizza un simbolo X ogni 3 caratteri letti – in questo modo viene calcolata la lunghezza del fattore w in unario.
2) Alla seconda passata la macchina copia il fattore w sul nastro 2, sfruttando il nastro 1 per vedere quando esso termina. Continua poi la lettura, controllando che il secondo e il terzo fattore w siano identici a quanto salvato nel nastro 2.

Complessità spaziale: vengono memorizzate 2 stringhe di lunghezza n/3, $S(n) = \Theta(n)$, con $n = |x|$.
Complessità temporale: la macchina effettua 2 scansioni lineari di x, $T(n) = \Theta(n)$.

Esercizio 4

È possibile definire una funzione ricorsiva che usa una visita in post-ordine per calcolare, per ogni nodo dell'albero, il numero di foglie e il massimo grado di sbilanciamento del sottoalbero radicato nel nodo. Facciamo uso di una struttura per memorizzare i due valori calcolati in ogni chiamata ricorsiva; restituiamo la struttura come valore di ritorno della funzione ricorsiva.

```
typedef struct {
        int leafs;
        int maxUnbalance;
        } Result;


Result unbalanceFactor(TREE T) {
Result Res, L, R;
 if (T == nil) {
        Res.leafs=0;
        Res.maxUnbalance=0;
        return Res;
 }
 if ((T.left == nil) and (T.right == nil )) {
        Res.leafs = 1;
        Res.maxUnbalance=0;
        return Res;
        }
 L = unbalanceFactor(T.left);
 R = unbalanceFactor(T.right);
 Res.leafs = L.leafs + R.leafs;
 Res.maxUnbalance = max(L.maxUnbalance, R.maxUnbalance,
                                abs(L.leafs − R.leafs));
 return Res;
}
```

La complessità dell'algoritmo è la stessa di una visita di un albero, cioè $O(n)$.