Algoritmi e Principi dell'Informatica

Appello del 2 Settembre 2015

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)

Si considerino i linguaggi seguenti:

$L_1 = \{ z \in \{a,b,c\}^* \mid$

$\forall x (\exists y (z = x.y) \rightarrow (\#a(x) \geq \#b(x) \geq \#c(x)) \wedge (\#a(x) - \#b(x) \leq 2) \}$

$L_2 = \{ z \in \{a,b,c\}^* \mid$

$\forall x (\exists y (z = x.y) \rightarrow (\#a(x) \geq \#b(x) \geq \#c(x)) \wedge (\#b(x) - \#c(x) \leq 2) \}$

$L_3 = \{ z \in \{a,b,c\}^* \mid$

$\forall x (\exists y (z = x.y) \rightarrow (\#a(x) \geq \#b(x) \geq \#c(x)) \wedge (\#a(x) - \#c(x) \leq 2) \}$

dove, per un carattere α e una stringa *x*, l'espressione $\#\alpha(x)$ denota il numero di volte in cui il carattere α si ripete nella stringa *x*. Ad es. #a(ababaccc) = 3.

Si costruisca una macchina astratta che riconosca $L = L_1 \cap L_2 \cap L_3$. Tra le diverse macchine astratte che riconoscono L sono preferite macchine "a potenza minima" ossia appartenenti alla categoria di automi a minor potenza riconoscitiva possibile.

Esercizio 2 (9 punti)

Sia $d : N \times N \rightarrow N$ la biiezione tra N e $N \times N$ definita come segue:

$$d(x,y) = (x+y)(x+y+1)/2 + x$$

Si considerino i seguenti insiemi:

- $S_1 = \{ d(x,y) \mid f_x(y) \neq \perp \}$
- $S_2 = \{ x \mid f_x(0) \neq \perp \}$
- $S_3(k) = \{ x \mid f_k(x) \neq \perp \}$, dove k è un parametro.

Per ciascuno dei predetti insiemi, si dica se esso è ricorsivo, motivando brevemente la risposta.

Esercizio 3 (8 punti)

Si definisca un algoritmo per determinare se due alberi binari di ricerca T1 e T2 sono uguali sia per il valore delle chiavi sia per la struttura. Valutare la complessità dell'algoritmo definito.

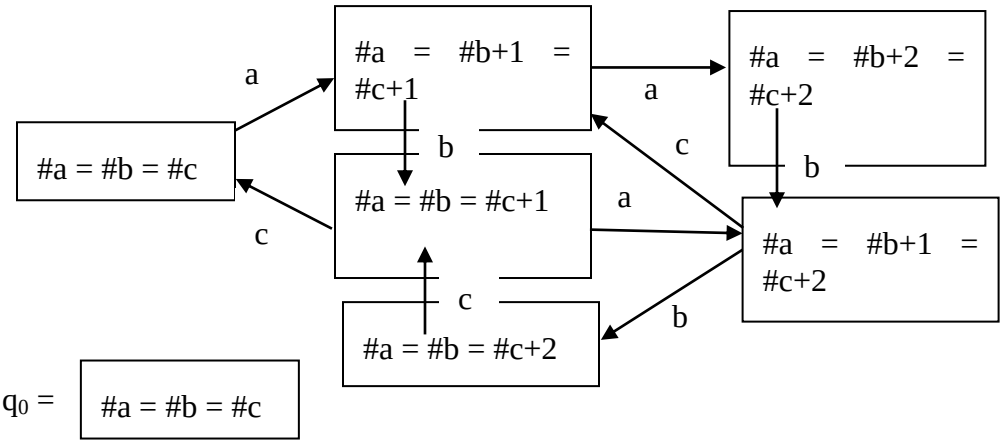
Esercizio 4 (punti 8)

Si consideri il seguente insieme di numeri interi:
{5, 60, 18, 23, 10, 15}.
Si individui una sequenza di inserimenti (senza cancellazioni) di questi valori come chiavi di un albero r-b (rosso-nero) inizialmente vuoto in modo che l'albero risultante abbia altezza complessiva doppia dell'altezza nera. Si ricorda che per convenzione la radice di un sottoalbero non viene computata nell'altezza (né nera né complessiva) mentre le foglie (i nodi T-NIL, neri) vengono computate.
Si mostri il risultato finale ottenuto dopo i vari inserimenti e almeno un risultato intermedio, ad esempio dopo l'inserimento dalla prima metà dei dati.

Tracce delle soluzioni

Esercizio 1

Poiché la differenza tra il numero di a, b e c in ogni prefisso delle stringhe del linguaggio è limitata, un automa a stati finiti è sufficiente per riconoscere il linguaggio:



F = Q

Esercizio 2

S_1 non è ricorsivo. Infatti la funzione d stabilisce banalmente una biiezione tra N e NxN, e se si potesse stabilire l'insieme delle coppie di numeri (x,y) tali per cui $f_x(y) \neq \perp$ allora il problema dell'arresto risulterebbe decidibile, il che è assurdo.

S_2 non è ricorsivo. Sia F l'insieme di funzioni calcolabili definite nel punto 0; tale insieme non è né l'insieme di tutte le funzioni computabili, né l'insieme vuoto. L'insieme S_2 di indici delle macchine di Turing che calcolino funzioni in F non è pertanto ricorsivo per il teorema di Rice.

La ricorsività di $S_3(k)$ dipende da k. Per alcuni valori di k si sa esattamente per quali valori di ingresso x la funzione $f_k(x)$ è definita. Ad esempio, se $f_k(x)$ fosse la funzione x^2, si saprebbe che è definita sempre e quindi il problema sarebbe decidibile (e deciso). Per altri valori di k, invece, il problema è indecidibile. Se, ad esempio, $f_k(x)$ fosse la funzione $f_x(0)$, tale funzione sarebbe certamente calcolabile, ma, come ad esempio osservato al punto b), non se ne potrebbe stabilire l'insieme di definizione.

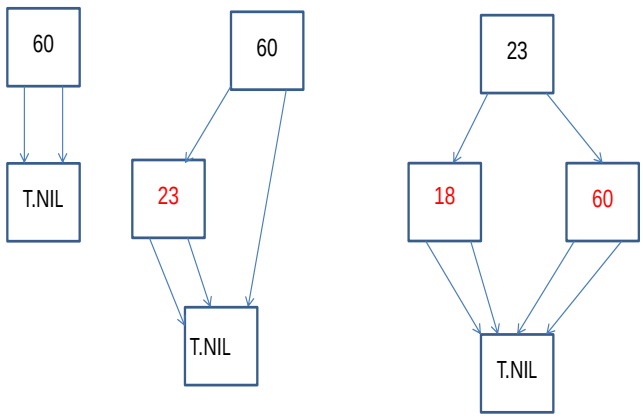
Esercizio 3

È possibile definire un algoritmo ricorsivo che, dopo aver verificato che le radici degli alberi non sono nulle e che le chiavi sono uguali, visita ricorsivamente il sottoalbero sinistro e quello destro e restituisce valore false se una delle condizioni verificate è falsa. La complessità dell'algoritmo è proporzionale al numero di nodi.

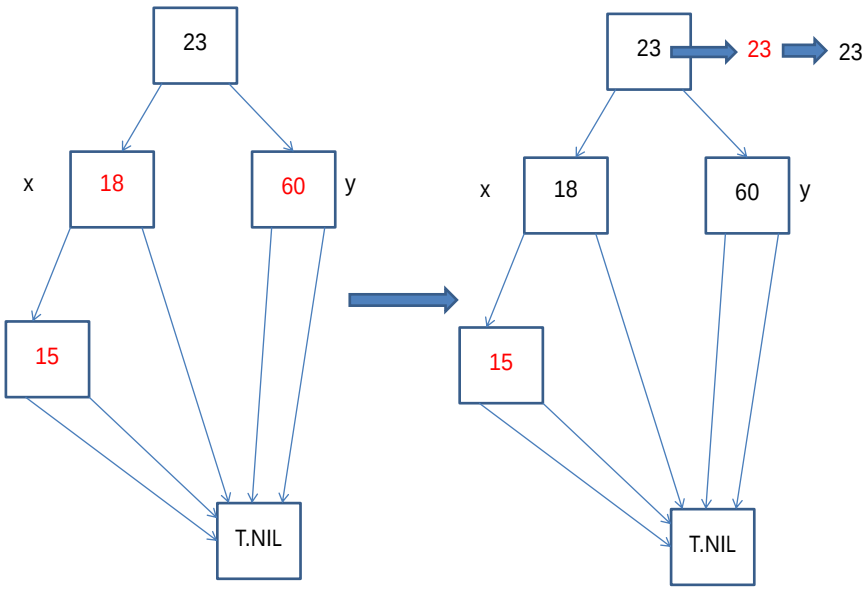
```
boolean compare(Tree T1, Tree T2){
if (T1 == null && T2 == null)
    return true;
if (T1 != null && T2 != null)
    return((T1.key == T2.key) && compare(T1.left, T2.left) &&
           compare(T1.right, T2.right));
  else return(false);
}
```

Esercizio 4

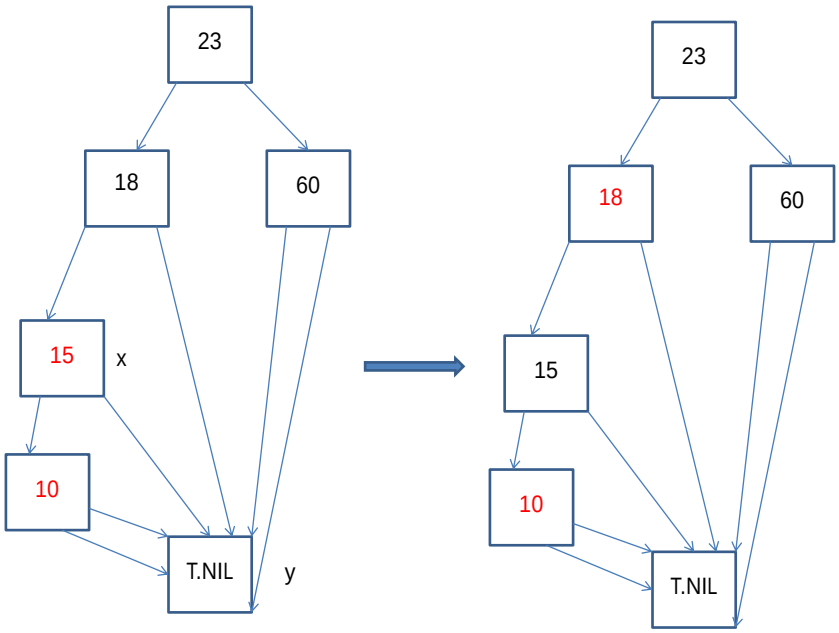
Se si inseriscono i dati in ordine crescente o decrescente (e.g. {60, 23, 18, 15,10, 5} si ottiene un albero in cui ogni sottoalbero ha la proprietà che il cammino sinistro alterna nodi rossi e nodi neri mentre il cammino destro ha solo nodi neri ed è quindi lungo la metà del sinistro. La figura seguente mostra lo stato dell'albero dopo l'inserimento di, rispettivamente:

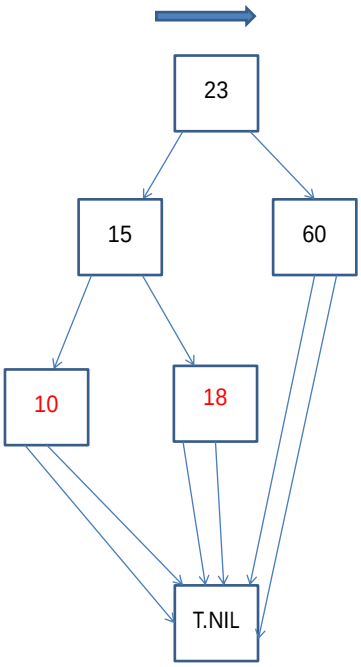


Inserimento di 60, 23, 18

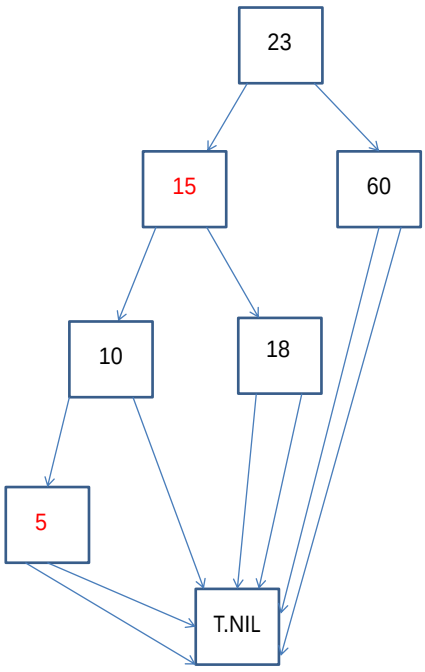


Inserimento di 15





Inserimento di 10



Risultato finale.