# Algoritmi e Principi dell'Informatica

**Appello del 10 Settembre 2012**

**Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 3 ore**

**Chi deve sostenere solo il modulo di Informatica teorica deve svolgere gli Esercizi 1, 2 e 3 in 1h e 45 min.**

**Chi deve sostenere solo il modulo di Informatica 3 deve svolgere gli Esercizi 3,4 e 5 in 1h e 45 min.**

**NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.**

**Esercizio 1 (punti 7/30-esimi)**

Si considerino le seguenti grammatiche e i linguaggi da esse generati; si indichi, per ognuna di esse, qual è il tipo di automa a potenza minima (all'interno delle "famiglie" tradizionali: a stati finiti, a pila deterministici, a pila nondeterministici, macchine di Turing) che riconosce il linguaggio.

1. $S \rightarrow a\ X\ b \mid b\ X\ a$

   $X \rightarrow a\ X\ b \mid b\ X\ a \mid c$

2. $S \rightarrow X \mid Y$

   $X \rightarrow a\ Y\ b \mid ab$

   $Y \rightarrow b\ X\ a \mid ba$

3. $S \rightarrow a\ X\ b \mid b\ Y\ a$

   $X \rightarrow a\ S\ b \mid ab$

   $Y \rightarrow b\ S\ a \mid ba$

4. $S \rightarrow X \mid Y$

   $X \rightarrow a\ Y\ b \mid b\ Y\ a$

   $Y \rightarrow b\ X\ a \mid a\ X\ b$

**Esercizio 2 (punti 8/30-esimi)**

Dire se le seguenti funzioni sono calcolabili, motivando adeguatamente la risposta:

1) f(x,y) è indefinita, se la y-esima MT (sia essa a nastro singolo) termina e durante tutta la computazione con ingresso x non esce mai dalle celle utilizzate per memorizzare la stringa x e le due celle contenenti blank immediatamente prima e dopo x, altrimenti è pari a 0;
2) g(z) = 1 se la funzione calcolata dalla z-esima MT è diversa dalla f del punto 1, 0 altrimenti.

**Esercizio 3 (punti 3/30-esimi)**

Si descriva a grandi linee una MT che accetti il Linguaggio 3 dell'Esercizio 1 e se ne valuti la complessità temporale. La descrizione della macchina può essere anche informale ma sufficientemente precisa da permettere la valutazione della complessità (asintotica) senza dubbi o ambiguità.

**Esercizio 4 (punti 7/30-esimi)**

Si considerino gli algoritmi `RB-INSERT` e `RB-DELETE` per l'inserimento e la cancellazione di nodi in e da alberi red-black. E' possibile, usando solo tali algoritmi, costruire un albero red-black fatto di esattamente 3 nodi, tutti neri?

Se sì, scrivere una sequenza di invocazioni di `RB-INSERT` e `RB-DELETE` che crea l'albero desiderato, motivando il perché la sequenza scritta produce la struttura richiesta.

**Esercizio 5 (punti 8/30-esimi)**

1. Definire un algoritmo che, dato in input un array A, restituisce il sottoarray [i,j] di A di lunghezza massima tale che, per ogni $i \leq k \leq j$, si ha $A[i] \leq A[k] \leq A[j]$.

Per esempio, nel caso dell'array [5, 18, 2, 12, 4, 7, 13, -2, -5, 12] il sottoarray cercato è [2, 12, 4, 7, 13].

2. Dire quale è la complessità temporale dell'algoritmo ideato.

**NB1:** L'algoritmo può essere descritto in pesudocodice, oppure anche in modo informale, purchè la descrizione sia sufficientemente precisa per determinare la complessità dell'algoritmo.

**NB2:** Il punteggio dato sarà tanto più alto quanto migliore sarà la complessità dell'algoritmo ideato.

**Soluzioni (parziali e schematiche)**

## Esercizio 1

1. Il linguaggio è costituito da stringhe di lunghezza dispari, con al centro una *c*, tali che se in una posizione prima della *c* si trova una *a* (rispettivamente, una *b*) allora dopo la *c*, in posizione simmetrica, si trova una *b* (rispettivamente, una *a*), e viceversa. Il linguaggio è riconoscibile da un semplice automa a pila deterministico, ma non da una automa a stati finiti.
2. Il linguaggio è composto da stringhe di lunghezza pari, con simboli *a* e *b* alternati, ed è regolare.
3. Il linguaggio è costituito da stringhe di lunghezza multipla di 4, che sono sequenze di coppie di tipo *aa* e *bb*, tali che se in una posizione prima del centro della stringa si trova una *a* (rispettivamente, una *b*) allora dopo il centro, in posizione simmetrica, si trova una *b* (rispettivamente, una *a*) , e viceversa. Il linguaggio è riconoscibile da un automa a pila non deterministico, perché uno deterministico non avrebbe la possibilità di identificare la posizione centrale.
4. Il linguaggio è vuoto, quindi regolare.

## Esercizio 2

1) f è calcolabile, infatti il numero di configurazioni da considerare per la MT y è limitato, quindi simulando la computazione possiamo controllare se si passa due volte per la stessa configurazione (loop) o se si esce dalla porzione di nastro fissata: in questo caso si restituisce 0. Se si arriva in fondo alla simulazione, si va in loop.

2) Non è calcolabile per il teorema di Rice: g è la funzione caratteristica dell'insieme degli indici di MT che non calcolano f, sicuramente né vuoto né coincidente con **N** (v. risposta del punto 1).

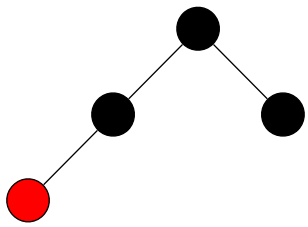
## Esercizio 3

Un MT che riconosca L3 può essere ottenuta mediante una semplice modifica della macchina che riconosce $\{ww^R\}$: un prima scansione copia la stringa di ingresso in un nastro di memoria; durante al scansione la macchina verifica anche che la stringa appartenga a {aa, bb}* e che abbia lunghezza multipla di 4; successivamente una delle due testine viene riposizionata all'inizio della stringa e facendo procedere quest'ultima da sinistra e destra e l'altra da destra a sinistra si verifica che quando una legge 'a' l'altra legga 'b' e viceversa.

La complessità temporale di una tale macchina è evidentemente $\Theta(n)$.

**Esercizio 4**

Per ottenere l'albero desiderato è sufficiente inserire 4 valori nell'albero. Questo crea un albero in cui i nodi ai primi 2 livelli sono neri, e sul terzo livello c'è un solo nodo di colore rosso, per esempio l'albero seguente (la posizione del nodo rosso dipende dai valori inseriti, comunque è poco rilevante):



Per ottenere l'albero desiderato, è a questo punto sufficiente cancellare, mediante `RB-DELETE`, il nodo rosso.

**Esercizio 5**

Un algoritmo possibile che risolve il problema dato in tempo lineare è il seguente:

```
MaxSubarray(A)
1  l_max := l := 1
2  u_max := u := 1
3  for i := 2 to A.length
4    if A[i] >= A[u]
5      u := i
6    elsif A[i] < A[l]
7      if u-l > u_max-l_max
8        l_max := l
9        u_max := u
10     l := u := i
11 if u-l > u_max - l_max
12   l_max := l
13   u_max := u
14 return <l_max,u_max>
```