# Algoritmi e Principi dell'Informatica

## Appello d'esame

17 luglio 2014

Nome:_____

Matricola:_____

Firma:_____

**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, 2 e 3 in 1 ora e 15 minuti.**

**Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 4 e 5 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.**

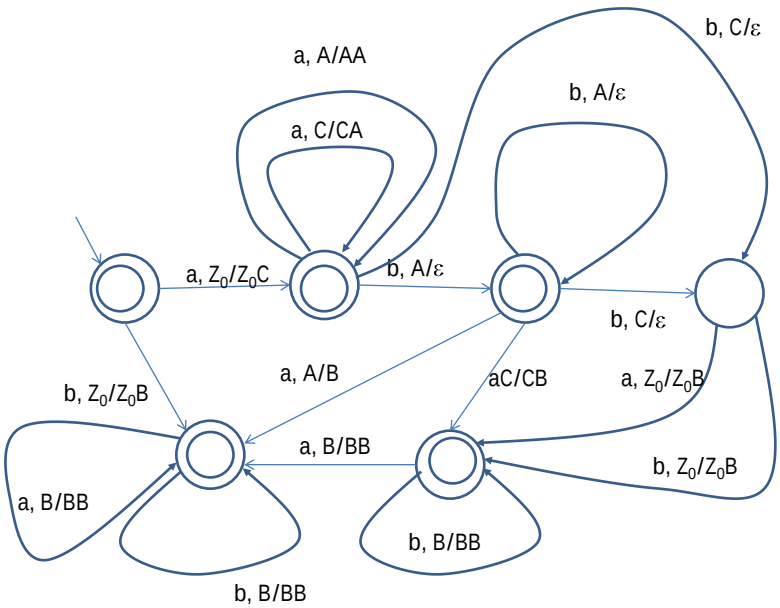
### Esercizio 1a (punti 5)

Si considerino la grammatica e l'automa a pila seguenti, e i linguaggi L(G) e L(A) definiti da essi sull'alfabeto {a,b}:

G:

$S \rightarrow \varepsilon \mid a\ S\ b$

A:




Si costruiscano un automa o una grammatica a potenza minima che riconoscano/generino rispettivamente $L(G) \cup L(A)$ e $L(G) \cap L(A)$.

L(G) è chiaramente il linguaggio $\{a^n b^n \mid n \geq 0\}$
A invece accetta tutte le stringhe non in L(G) oltre la stringa nulla.
Quindi $L(G) \cup L(A) = \{a, b\}^*$ e $L(G) \cap L(A = \{\varepsilon\}$, entrambi linguaggi regolari.

**Esercizio 2 (punti 6)**

Si consideri la seguente formula F:

$\forall x \, \forall y \, ( \exists z \, m(x,y,z) \rightarrow \exists t \, m(x,y,t) )$

**2.a**

Come noto, una formula può essere intesa in senso puramente sintattico come una sequenza di simboli. Si indichi la categoria sintattica (funzione, predicato, ecc.) di ciascun simbolo utilizzato in F; se pertinente, indicare anche la *arietà* associata al simbolo.

Si dica anche se F è una formula chiusa.

$\forall$ quantificatore universale
x variabile
z variabile
$\exists$ quantificatore esistenziale
y variabile
m predicato con arietà 3
( punteggiatura
, punteggiatura
) punteggiatura
$\rightarrow$ connettivo logico (implicazione)
t variabile

La formula è chiusa in quanto tutte le variabili sono quantificate.

**2.b**

Si consideri poi un'interpretazione in cui m(x,y,z) indica il fatto che la macchina di Turing con indice y e ingresso x produce z in uscita.

E' vera la formula F sotto l'interpretazione data? Motivare la risposta.

La formula F è vera nell'interpretazione data.
Se, data una coppia di valori <x,y>, esiste un valore z tale che <x,y,z> soddisfi m (ossia la funzione calcolata dalla macchina di Turing con indice y per l'ingresso x è definita e restituisce z), tale tripla può essere utilizzata per soddisfare anche la parte a destra dell'implicazione (basta scegliere un valore di t uguale a z).
Se invece non esiste un valore di z tale che <x,y,z> soddisfi m (ossia la funzione calcolata dalla macchina di Turing con indice y per l'ingresso x non è definita), l'implicazione è banalmente soddisfatta poiché l'antecedente è falso.

**Esercizio 3 (punti 6)**

Il signor Giovanni sta cercando di procurarsi un compilatore "open source" o commerciale che traduca codice C nell'assembler del suo hardware xyz. Egli sa che non tutto il SW open-source o disponibile sul mercato è affidabile al 100% (eufemismo). Quindi, prima di procedere all'acquisizione (che intende utilizzare per sviluppare applicazioni altamente critiche) intende anche e preliminarmente acquisire sufficienti "garanzie" (intese in senso tecnico, non semplicemente commerciale (in caso di danni previsione di adeguato risarcimento)). Quindi, prima ancora di mettersi alla caccia del compilatore che faccia al caso suo si mette alla ricerca di uno strumento e/o di un benchmark che gli possa verificare con certezza il corretto funzionamento del compilatore che intende esaminare per l'eventuale acquisizione.

Può la ricerca del signor Giovanni (quella dello strumento o del benchmark, non quella successiva del compilatore) avere successo? Si spieghi brevemente il motivo della risposta.

Il problema della correttezza di un compilatore, come la correttezza di un qualsiasi programma è formalizzabile come il problema di stabilire se un generico algoritmo calcola una funzione data; quindi in tal caso è applicabile il teorema di Rice poiché la compilazione dal C a un linguaggio assembler è formalizzabile come una specifica funzione computabile (diversa dall'insieme vuoto e dall'insieme universo di tutte le funzioni computabili).
Nessun benchmark può quindi certificare la correttezza assoluta (assenza di errori) di un programma (a meno di non ridurre il problema a un dominio finito e ipotizzare un testing esaustivo).

**Parte opzionale**

Che cosa consiglierete al signor Giovanni riguardo alla sua ricerca: è preferibile orientarsi verso un opportuno benchmark oppure verso uno strumento che certifichi la correttezza del compilatore?

Un benchmark ben congegnato ha buone possibilità, se il programma testato è scorretto, di evidenziarne eventuali errori (semidecidibilità della Scorrettezza di un programma). E' quindi preferibile consigliare al signor Giovanni la ricerca di un buon benchmark piuttosto che quella di un tool che per motivi teorici non potrà mai garantire la correttezza di un qualsiasi compilatore.
Tuttavia, ciò non esclude che il signor Giovanni possa trovare su Internet un compilatore corredato di una specifica certificazione di correttezza che matematicamente DIMOSTRI tale correttezza come risultato di un'adeguata analisi formale eseguita da esperti del settore (certificazione che a sua volta Giovanni potrebbe controllare se fornito di adeguata competenza teorica o fare verificare da un tool automatico per la verifica di correttezza di prove matematiche).

# Algoritmi e Principi dell'Informatica

## Appello d'esame

17 luglio 2014

Nome:_____

Matricola:_____

Firma:_____

**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, 2 e 3 in 1 ora e 15 minuti.**

**Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 4 e 5 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 4 (punti 7)**

Dato il seguente algoritmo che calcola la reverse di una coda di elementi

```
Reverse(P):
    If queue P is not empty do:
        element  := Dequeue (P)
        Reverse (P)
        Enqueue (P, element)
```

Ricavare e risolvere in maniera precisa l'equazione alle ricorrenze

T(n) = ?

(si osservi che il tempo per fare il reverse di una coda di 1 elemento , T(1), è costante)

**Condizione iniziale:** il tempo per fare il reverse di una coda di un elemento è costante
T(1) = O(1) = 1

**Relazione alle ricorrenze**: il tempo per fare il reverse di una coda di N elementi è il tempo di fare il reverse di una coda dì N-1 elementi più 2 operazioni (dequeue e enqueue).

T(N) = T(N-1) + 2
Sostituendo abbiamo
T(N-1) = T(N-2) + 2
T(N-2) = T(N-3) + 2
. . . . . .
T(2) = T(1) + 2
T(1) = 1

sommiamo a sinistra e a destra:

$T(N) + T(N-1) + T(N-2) + T(N-3) + \ldots T(3) + T(2) =$

$= T(N-1) + T(N-2) + T(N-3) + \ldots T(3) + T(2) + T(1) + 2*(N-1)$

semplificando :

$T(N) = T(1) + 2*(N-1) = 1 + 2*(N-1)$

l'algoritmo è quindi **O(N)**

**Esercizio 5 (punti 9)**

Si considerino due alberi binari di ricerca (BST). Si definisca un algoritmo per costruire un BST contenente i dati di entrambi gli alberi e che sia "ragionevolmente" bilanciato, ossia abbia un'altezza dello stesso ordine di grandezza di quella di un albero perfettamente bilanciato. Se ne valutino le complessità spaziale e temporale.

1) Si effettua visita "in ordine" dei due alberi T1 e T2, ottenendo due liste ordinate L1 e L2 (complessità $\Theta(n1+n2)$, dove n1 = |L1|, n2 = |L2|).
2) Si fondono le due liste in un array ordinato ($\Theta$ (n1+n2)).
3) Si crea un BST partendo dall'elemento centrale dell'array, poi ricorrendo sui due sottoarray rimanenti ($\Theta$ (n1+n2)).
Quindi le complessità risultanti sono entrambe lineari nel numero dei nodi presenti nei sottoalberi: $S(n) = T(n) = \Theta(n)$, dove n = n1+n2.