# Algoritmi e Principi dell'Informatica

**Appello del 2 Marzo 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)

Una macchina di Turing si dice *n*-generatrice se, quando riceve 0 in ingresso, termina entro *n*+1 passi scrivendo il numero *n* in uscita ($n \geq 0$).

Fissata una qualsiasi enumerazione algoritmica $\mathcal{E}$ delle macchine di Turing, il *generatore minimo* di un numero *n*, indicato con *gm*(*n*), è il più piccolo indice, secondo $\mathcal{E}$, di macchina di Turing *n*-generatrice. Si risponda alle seguenti domande fornendo brevi ma chiare spiegazioni per le risposte date:

a) La funzione *gm* è totale?
b) La funzione *gm* è computabile?

## Esercizio 2 (punti 9)

Si consideri il seguente linguaggio:

$L = \{a^n b^{3n} \mid n \geq 0\} \cup \{a^n (bbbc)^n \mid n \geq 0\}$

Si forniscano:

a) Una grammatica, preferibilmente a potenza generativa minima, che generi L
b) Un automa, preferibilmente a potenza minima tra le famiglie tradizionali a stati finiti, a pila (deterministici o no), macchine di Turing, che riconosca L
c) Una rete di Petri che riconosca L

**Esercizio 3 (punti 9)**
Si consideri il seguente linguaggio:
$L = \{a^{n_1}ba^{n_2}b\ldots a^{n_k} \mid 1 \leq k \leq 100, \forall\ 1 \leq i \leq k.(n_i \geq 1), \exists\ 1 \leq i < j \leq k.(n_i = n_j)\}$
Si descrivano a grandi linee, ma in maniera sufficientemente precisa, una MT e una RAM che riconoscano L e se ne valutino le complessità temporali, nel caso della RAM sia a criterio di costo costante che a criterio logaritmico.
Sono ovviamente preferite soluzioni a minor complessità per entrambi i tipi di macchina.

**Esercizio 4 (punti 8)**
1.
Si scriva un algoritmo che, data in input una sequenza *P* contenente i dati anagrafici di diverse persone, determina se nella sequenza ci sono 2 persone che compiono gli anni lo stesso giorno dell'anno. Si assuma pure che i dati di ogni persona siano rappresentati da un oggetto che ha diversi attributi, *nome*, *indirizzo*, *codice_fiscale*, *data_nascita*, e che la data di nascita a sua volta sia un oggetto con 3 attributi, *giorno*, *mese*, *anno* (per cui se *p* è il riferimento ad una persona, *p.data_nascita* è la sua data di nascita, e *p.data_nascita.giorno* è il giorno del mese in cui la persona è nata).
Si dia la complessità temporale asintotica dell'algoritmo scritto nel caso pessimo.

2.
Come al punto 1: Come cambia, se cambia, la complessità dell'algoritmo se, sapendo che tutte le persone della sequenza *P* sono nate nel XX secolo, occorre determinare se nella sequenza *P* ci sono 2 persone nate esattamente lo stesso giorno?

3.
Come cambia, se cambia, la complessità dell'algoritmo se, sapendo che tutte le persone della sequenza *P* sono viventi, occorre determinare se nella sequenza *P* ci sono 2 persone nate esattamente lo stesso giorno?

**NB:** Il punteggio dato sarà tanto maggiore quanto migliore è la complessità degli algoritmi scritti.

**Tracce delle soluzioni**

**Esercizio 1**

a) La funzione è certamente totale, in quanto, per qualunque numero $n$, si può facilmente costruire una macchina di Turing $n$-generatrice. Basta, infatti, compiere un passo di stampa per ogni cifra di $n$ (per un totale di un passo se $n$=0 e di $\lfloor \log_2 n \rfloor$+1 passi se $n$>0, quindi sempre entro $n$+1 passi). Tale macchina potrebbe non essere la $n$-generatrice con l'indice più piccolo, ma la sua esistenza implica che la funzione $gm$ è definita nel punto $n$.

b) Sì. Infatti, per quanto osservato al punto a), si può facilmente trovare un limite superiore al valore di $gm(n)$. A tale scopo, sia $i$ l'indice della macchina esibita al punto a). Basta a questo punto, mediante una macchina di Turing universale, testare, in successione, ciascuna delle macchine di Turing con indice da 0 a $i$ per $n$+1 passi a fronte dell'ingresso 0: il primo indice per il quale la macchina termina e produce $n$ in uscita è il valore $gm(n)$ cercato.

**Esercizio 2**

## Punto a.

$S \rightarrow X \mid Y \mid \varepsilon$

$X \rightarrow aXbbb \mid abbb$

$Y \rightarrow aYbbbc \mid abbbc$

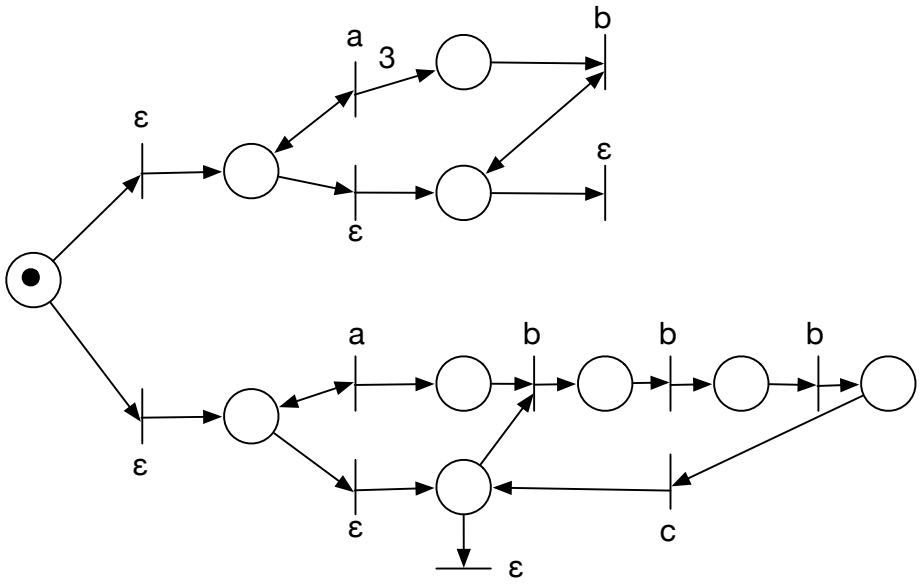
## Punto b.

Un automa a pila *deterministico* A che riconosca L può essere costruito secondo le linee seguenti:

1. A impila gli n a (ponendo lo stato iniziale tra quelli finali, accetta anche la stringa vuota);
2. Al ricevimento del primo b ne conta altri due;
3. Dopo aver letto i primi 3 b, si sposta in due stati diversi a seconda che il carattere successivo sia b o c (e ovviamente va in uno stato di errore se non siverifica nessuna delle circostanze previste)
4. Da questo punto in poi verifica che per ogni a impilato si trovino 3 b oppure 3 b seguite da c a seconda dello stato scelto al punto precedente.

## Punto c.

La rete di Petri seguente (con la relativa marcatura iniziale) riconosce L quando viene a trovarsi con tutti i posti vuoti.




**Esercizio 3**

**Soluzione mediante MT**

Grazie al fatto che il parametro k è limitato e ≤ 100 è possibile progettare diverse MT che riconoscano L con complessità temporale O(n).

Un primo modo consiste nel costruire una MT con 100 nastri di memoria: ogni sequenza $a^{ni}$ viene memorizzata su un nastro diverso; successivamente tutte le testine vengono riposizionate sulla prima cella dei rispettivi nastri e vengono spostate di una posizione a destra contemporaneamente: se due di essere giungono alla fine della propria sottostringa contemporaneamente (ciò può essere verificato mediante la funzione δ della macchina che produce uno stato di accettazione se due testine leggono contemporaneamente una marca di fine stringa) la stringa originaria viene accettata.

Una MT con meno nastri invece memorizza la stringa in ingresso in due nastri di memoria e, per ogni frammento $a^{n_i}$ di un nastro verifica se sull'altro nastro ne esiste un altro con lo stesso numero di a. Si noti che per evitare di considerare il caso $n_i = n_j$ quando i = j può essere opportuno marcare in qualche modo la porzione di nastro sotto esame, e.g. sostituendo la b iniziale con un altro carattere. La MT prende in considerazione il primo gruppo $a^{n_1}$ dal primo nastro e scorre il secondo nastro per verificare se esiste un altro gruppo con lo stesso numero di a (per fare questo deve fare k volte andata e ritorno nello scorrimento di $a^{n_1}$ impiegando un tempo $2.k.n_1 + n$; il processo viene ripetuto per tutti gli $a^{n_i}$ del primo nastro ossia k volte; si ha quindi una complessità quadratica in k ma ancora lineare in O(n).

**Soluzione mediante RAM**

Una RAM può procedere con uno schema algoritmico uguale a quello utilizzato per la terza MT descritta sopra. A criterio di costo costante ciò richiede O(n) per la prima fase e $O(k^2)$ per la seconda; essendo k ≤ 100, il costo totale è O(n).

A criterio di costo logaritmico invece la prima fase costa O(n.log(n)) ma la seconda $O((k.\log(n))^2)$ ossia ancora $O(\log(n)^2)$, comunque dominata dalla fase precedente.

**Esercizio 4**

1.

In alternativa al confronto tra tutti gli elementi della sequenza (complessità $\Theta(n^2)$, n = lunghezza della sequenza *P*) è possibile usare un vettore di appoggio di dimensione pari al numero di giorni dell'anno (365) e conteggiare così la frequenza in *P* di ogni singolo giorno dell'anno. Questa soluzione avrebbe complessità temporale $\Theta(n)$. L'uso del vettore di appoggio comporta un incremento costante, quindi trascurabile, della complessità spaziale.

È possibile tuttavia considerare che, poiché al massimo ci sono 356 giorni in un anno (contando la possibilità di anni bisestili), se l'array *P* è più lungo di 356 elementi di sicuro ci sono almeno 2 persone che compiono gli anni lo stesso giorno. Se ce ne sono meno di 356, è accettabile anche eseguire un doppio ciclo for, in quanto comunque il ciclo è ripetuto un numero limitato di volte.

```
stesso_compleanno (P)
1  if P.length > 356
2    return true
3  for i := 1 to P.length -1
4    for j := i+1 to P.length
5      if P[i].data_nascita.giorno = P[j].data_nascita.giorno and
          P[i].data_nascita.mese = P[j].data_nascita.mese
6        return true
7  return false
```

Ognuno dei 2 cicli annidati 3-4 viene eseguito al più 356 volte, quindi la complessità dell'algoritmo è $\Theta(1)$.

2.

Anche in questo caso se *P* contiene più elementi di un certo numero K costante fissato di sicuro ci sono 2 persone che sono nate lo stesso giorno. In questo caso K = 356*100 (il valore è anche minore, in quanto non tutti gli anni sono bisestili, ma non cambia niente dal punto di vista della complessità), quindi alla riga 1 occorre cambiare 356 in 35600, ma comunque la complessità rimane costante (e aggiungere il controllo sull'anno alla riga 5).

3.

In questo caso, considerando che un uomo non vive fino a 130 anni, basta prendere K=356*130, ed il ragionamento del punto 2 rimane invariato. (Si noti che non cambia nulla se, per stare larghi, si prende K=356*200).