Informatica Teorica
Esame del 1/9/2009 Durata: 2h

Esercizio 1 (punti 9)
Si consideri la seguente funzione: $g : N \rightarrow R$, $g(x) = \sqrt{x}$.
1) Dire se la funzione g è totale. Giustificare la risposta.
Si consideri una implementazione di tale funzione, per esempio una macchina di Turing M, che, ricevendo in ingresso il numero naturale x rappresentato in decimale, scrive in uscita la sua radice quadrata, sempre in formato decimale.
2) M calcola una funzione totale? Giustificare la risposta.
3) Sia m il numero di Goedel di M. Il problema $m \in \{y \mid f_y(x) \neq \perp, \forall x \in N\}$ è computabile? In caso positivo, fornire una MT che risolva il problema; in caso negativo, motivare la risposta.

Esercizio 2 (punti 12)
Si consideri il seguente linguaggio:
$L1 = \{w_1.y.w_2.y.w_3 \in \{0, 1\}^* \mid |y| = 2\}$.
1) Si definisca un automa (a potenza minima) che riconosca L1.
2) Si definisca una grammatica (senza nessun vincolo sulla potenza espressiva) che generi L1, possibilmente usando un numero minimo di nonterminali.
Si consideri ora il linguaggio:
$L2 = \{w_1.y.w_2.y.w_3 \in \{0, 1\}^* \mid \exists k > 0 : |y| = 2k\}$.
3) Qual'è la relazione tra L1 e L2? Si motivi adeguatamente la risposta.
4) Cambiano (in caso positivo dire come) le risposte ai punti 1 e 2 nel caso di L2? Si motivi adeguatamente la risposta.

Esercizio 3 (punti 12)
Il seguente programma, scritto in un linguaggio simil-C, calcola il valore del coefficiente binomiale $\binom{n}{k}$, di cui si ricorda la definizione: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, memorizzandolo nella variabile *c*.

```
nf = 1;
for i = 2 to n do nf = nf * i;
kf = 1;
for i = 2 to k do kf = kf * i;
nkf = 1;
for i = 2 to n-k do nkf = nkf * i;
c = nf / (kf * nkf);
```

1) Si valuti la complessità spaziale e temporale dell'algoritmo, adottando sia il criterio a costo costante che logaritmico. NB: è possibile per semplicità indicare anche solo la *classe* di complessità, motivando però adeguatamente la risposta.
(Suggerimento: si usi l'approssimazione di Stirling del fattoriale: $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$)

Il seguente frammento di codice implementa una variante più efficiente dell'algoritmo. La variabile *c* viene tutt'ora usata per memorizzare $\binom{n}{k}$, riducendo però il numero delle motiplicazioni eseguite. Per es., calcola $\binom{13}{4} = \frac{13!}{4!9!}$ come $\binom{13}{4} = \frac{13 \cdot 12 \cdot 11 \cdot 10}{4 \cdot 3 \cdot 2 \cdot 1}$

```
c = 1;
m = min(k, n-k),       /* min is the minimum operator */
cnum = n – m + 1;
cden = m;
for i = 1 to m do
  c = c * cnum / cden;
  cnum = cnum + 1;
  cden = cden – 1;
end;
```

2) Si indichi qual'è il caso pessimo per la complessità, per un certo valore fissato di *n.*
3) Si valuti la complessità *spaziale* con criterio logaritmico (caso pessimo), e si dica se essa sia cambiata come classe di complessità rispetto alla prima implementazione.

SOLUZIONI

Es 1

1) sì
2) no, per esempio i numeri irrazionali hanno una rappresentazione decimale che necessita di un numero infinito di cifre
3) (e 4): sì, è computabile, infatti la risposta alla domanda è no e corrisponde alla MT che calcola la funzione costante 0.

Es 2

Automa a stati finiti (nondeterministico per comodità):

Stato	0	1
init	{init, q0}	{init, q1}
q0	q00	q01
q1	q10	q11
q00	{q00, r0}	q00
q01	{q01, r1}	q01
q10	q10	{q10, r0}
q11	q11	{q11, r1}
r0	end	-
r1	-	end
end	end	end

L'automa si basa sulla memorizzazione negli stati q00, q01, q10, q11 del contenuto della prima y (per es. se siamo in q10, allora y=10). Viene poi controllato che ci sia una successiva sottostringa y'=y.

Grammatica:
S -> W00W00W | W01W01W | W10W10W | W11W11W
W -> 0W | 1W | ε

L1 ed L2 sono lo stesso linguaggio. Infatti se due stringhe sono uguali, allora tutti i loro prefissi di uguale lunghezza sono uguali (in questo caso di lunghezza 2).

Es 3

1. With the uniform criterion the time complexity is Θ(n) and the space complexity is constant. With the logarithmic cost, the dominant component it the one concerning the computation of *n*!, whose time complexity is $\Theta(n^2 \cdot \log n)$, and space complexity is $\Theta(n \cdot \log n)$.
2. The worst case corresponds to the highest value of the coefficient, occurring with k=n/2, when the value of the coefficient is $\binom{n}{(n/2)} = \frac{n!}{((n/2)!)^2}$.
3. The space complexity is determined by the greatest value computed in variable *c*, which is $\frac{n!}{((n/2)!)^2}$.

   The size of the binary string encoding this value is
   $l\left(\frac{n!}{((n/2)!)^2}\right) = \log(n!) - 2\log\left(\left(\frac{n}{2}\right)!\right) \approx n \cdot \log(n) - 2 \cdot \frac{n}{2} \cdot \log\frac{n}{2} = n \cdot \log(n) - n \cdot \log\left(\frac{n}{2}\right) = n$ which is Θ(n),
   hence the space complexity is changed.