Algoritmi e Principi dell'Informatica

Appello del 15 Settembre 2016

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 (8 punti)

In matematica, si definiscono *numeri primi gemelli* due numeri primi che differiscano tra loro di 2.

La *congettura dei numeri primi gemelli* afferma che esistono infinite coppie di numeri primi gemelli.

Specificare tale congettura mediante una formula di logica del prim'ordine. Nella specifica è consentito utilizzare:

- come funzioni, le usuali quattro operazioni aritmetiche;
- come costanti, i numeri naturali 0, 1, 2, …;
- come predicati, il predicato di uguaglianza (=), i comparatori (<, >, ≥, ≤) e un predicato unario *nat* che indica che il suo argomento è un numero naturale.

È inoltre possibile, e consigliabile, specificare dei predicati ausiliari per render più agevole la scrittura e la comprensione della formula; tuttavia la semantica di tali predicati dovrà essere esplicitamente definita mediante opportune formule che facciano riferimento ai simboli base di cui sopra.

Esercizio 2 (9 punti)

Punto a)

Ad oggi, la più grande coppia di numeri primi gemelli trovata è $3756801695685 \cdot 2^{666669} \pm 1$. Si tratta di numeri con ben 200700 cifre, ma non è ancora stato dimostrato l'enunciato della congettura, che potrebbe quindi anche essere falsa.

Che cosa si può concludere in merito alla decidibilità dell'enunciato della congettura?

Punto b)

Sia f una funzione definita come segue:

$f(x) = 1$ se esiste una coppia di numeri primi gemelli maggiori di x;

$f(x) = 0$ altrimenti.

La funzione f è calcolabile?

Esercizio 3 (9 punti)

Si consideri il linguaggio L_q di figure definito sull' alfabeto {a, b}, composto da tutte e sole le figure *quadrate* fatte da righe di soli simboli 'a' alternate a righe di soli simboli 'b', partendo con le 'a'.

Es.
```
aaaa
bbbb
aaaa
bbbb
```

Si descrivano esaurientemente due macchine di Turing che accettino L_q, una a k-nastri e una a nastro singolo, sapendo che la figura è fornita sul nastro in ingresso riga per riga e senza separatori, e se ne studino le complessità *spaziali e temporali*.

Esercizio 4 (7 punti)

Sia dato un vettore di interi, *A*, ordinato in ordine non decrescente. Sia dato inoltre un valore intero *v*. Si definisca un algoritmo che verifichi se *A* contiene due elementi a_i e a_j tali che $a_i - a_j = v$. Il punteggio dell'esercizio sarà commisurato alla complessità temporale dell'algoritmo.

Tracce delle soluzioni

Esercizio 1

Definisco per comodità un predicato che cattura il concetto di numero primo:

$\forall x\ (primo(x) \leftrightarrow nat(x) \wedge \neg\exists y \exists z\ x = y \cdot z \wedge nat(y) \wedge nat(z) \wedge y>1 \wedge z>1)$

La congettura è quindi specificata come segue:

$\forall x\ (nat(x) \rightarrow \exists y \exists z\ y>x \wedge primo(y) \wedge z=y+2 \wedge primo(z))$

Esercizio 2

Punto a)

Si tratta di una questione certamente decidibile poiché la formula che la specifica è chiusa.

Punto b)

Se la congettura dei numeri primi gemelli è vera, allora la funzione *f* vale costantemente 1, ed è pertanto calcolabile.

Se la congettura è falsa, allora esiste un numero *k* oltre il quale non è più vero che esistono due numeri primi gemelli (e prima del quale esistono). Pertanto la funzione *f* sarà fatta a "scalino", e sarà pari a 1 per ingressi minori di *k*, e pari a 0 per ingressi maggiori o uguali a *k*. Anche in questo caso la funzione è certamente calcolabile.

Esercizio 3

Macchina a k-nastri:

si legge la prima sequenza di 'a' copiandole su 2 nastri di memoria; poi il primo nastro viene utilizzato per contare il numero di elementi di ogni riga, mentre il secondo serve per contare il numero di righe.

Chiaramente basta una passata sulla stringa in ingresso. Siano m gli elementi di una riga, la stringa è lunga $n = m^2$, mentre la memoria occupata risulta 2m.

Quindi: $S(n) = \Theta(\sqrt{n})$, $T(n) = \Theta(n)$.

Macchina a nastro singolo:

l'idea è controllare la lunghezza di ogni riga, cambiando di volta in volta le 'a' in 'A' e le 'b' in 'B' riga per riga, da sinistra verso destra, a parte i primi elementi di ogni riga che verranno cambiati in 'X'. In questo modo possiamo controllare che tutte le righe siano lunghe uguali, facendo $\Theta(nm)$ mosse.

Per controllare che il numero di righe sia corretto, la macchina può controllare che il numero delle 'X' sia uguale al numero di 'A' della prima riga + 1, facendo ulteriori $\Theta(nm)$ mosse (ad esempio cambiandoli uno alla volta in altri simboli).

Quindi: $S(n) = \Theta(n)$, $T(n) = \Theta(n\sqrt{n})$.

Esercizio 4

Una soluzione banale, ma poco efficiente, controlla tutte le coppie, *A[i]*, *A[j]*, con $1 \leq i < j \leq$ *A.lenght*, per verificare se *A[j] - A[i] = v*. La complessità di tale soluzione è $O(n^2)$.

È possibile poi definire una soluzione con complessità O(n log n). Per ognuno degli *n* elementi dell'array, A[i], si applica una *ricerca binaria* agli elementi successivi per cercare un elemento pari a v+A[i].

```
CercaCoppia-RicercaBinaria (A, v) {
for i = 1 to A.length
      if (RicercaBinaria(A[i + 1 . . . A.length], v + A[i]))
            return true;
 return false;
}
```

È infine possibile ottenere una soluzione più efficiente, di complessità O(n), scandendo gli elementi in modo lineare tramite un opportuno incremento di due indici, i e j:

```
CercaCoppia-Lineare (A, v) {
i = 1;
j = 2;
while (j ≤ A.length)
      if ((A[j] – A[i]) < v)
            j = j + 1;
      else if ((A[j] – A[i]) > v)
            i = i + 1;
                        else return true;
return false;
}
```

A ogni iterazione, si incrementa j oppure i - oppure si esce dalla funzione. A ogni iterazione $j \geq i$; si eseguono quindi O(n) iterazioni.