Algoritmi e Principi dell'Informatica

Appello del 14 Luglio 2017

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 uno spazio a d dimensioni, un punto p domina un punto q se

- in ogni dimensione, la coordinata di p è maggiore o uguale a quella corrispondente di q;
- c'è almeno una dimensione in cui la coordinata di p è strettamente maggiore di quella di q.

Ad esempio, in 3 dimensioni, il punto p=(1,2,3) domina il punto q=(0,2,1), mentre non domina il punto r=(0,1,4).

E' già disponibile una funzione binaria *elem* che, con argomenti a (array) e i (indice), restituisce l'(i+1)-esimo elemento dell'array a (il primo elemento ha indice 0). Si conviene inoltre che, per un array a di k elementi, sia $elem(a,i)=\perp$ per $i \geq k$.

Un punto d-dimensionale viene rappresentato come un array di d elementi, cui si può accedere tramite la suddetta funzione *elem*. Oltre a *elem*, si possono considerare già disponibili i comparatori ($>$, $\geq$, $=$) e il simbolo $\perp$.

a) Si specifichino in logica del prim'ordine i seguenti predicati binari:
   - *stessaDim*, che indica che i suoi due argomenti sono due punti di uguale dimensionalità;
   - *uguale*, che indica che i suoi due argomenti sono due punti uguali;
   - *domina*, che indica che il primo argomento è un punto che domina il secondo argomento.

Il programma SKYLINE riceve in ingresso un insieme P di punti tutti diversi e tutti con la stessa dimensionalità, e restituisce in uscita S, la cosiddetta "skyline" di P, ossia l'insieme dei punti di P che non sono dominati da nessun altro punto di P.

b) Si utilizzi la notazione di Hoare per specificare opportune pre-condizioni e post-condizioni per il programma SKYLINE. Si possono considerare già disponibili unicamente i predicati definiti al punto a) nonché il simbolo di appartenenza insiemistica ($\in$) applicato a P e a S.

Esercizio 2 (8 punti)

a. Si considerino le famiglie $\mathcal{M}_1$ e $\mathcal{M}_2$ di macchine di Turing (MT) che calcolano *funzioni totali* definite sui numeri naturali e *periodiche*, rispettivamente di periodo T_1 e T_2. E' decidibile il problema di stabilire se, date due MT appartenenti, rispettivamente a $\mathcal{M}_1$ e $\mathcal{M}_2$ esse siano equivalenti?

b. Si consideri la famiglia $\mathcal{M}^2$ di MT deterministiche che hanno complessità spaziale $S_M(n) = n^2$ dove n è la lunghezza della stringa in ingresso[1]. E' decidibile il problema di stabilire se, data una qualsiasi MT $M \in \mathcal{M}^2$ e una stringa x, $x \in L(M)$?

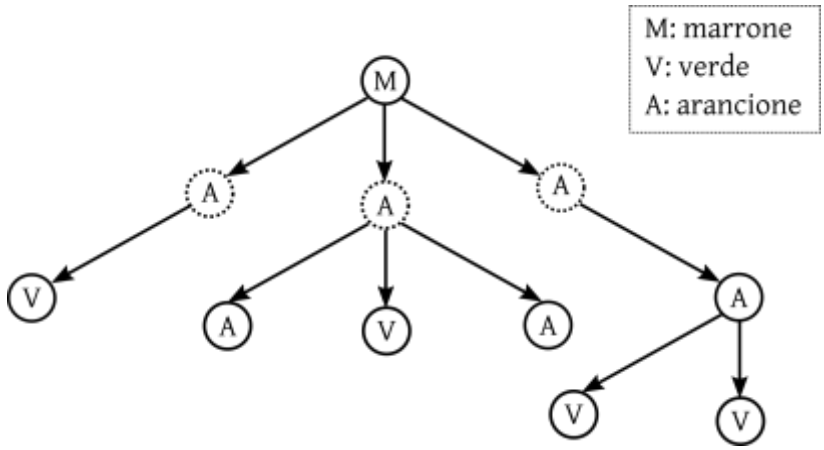
---

[1] Per chi non avesse ancora seguito la parte "informatica 3" del corso, si ricorda che un MT ha complessità spaziale $S_M(n) = n^2$ se durante il funzionamento su una stringa di ingresso lunga n, non occupa mai più di n^2 celle di memoria (escludendo le celle del nastro di ingresso).

c. **(parte opzionale; valida per ulteriori punti 3; sarà valutata solo se i precedenti punti a. e b. avranno ricevuto risposte corrette)**
Cambia la risposta alla domanda **b.** se le MT in questione sono nondeterministiche? NB: si assuma che la complessità spaziale di una MT nondeterministica sia $S_M(n) = n^2$ se *tutte le sue possibili computazioni* su un ingresso di lunghezza n non occupano più di n^2 celle di memoria.

Esercizio 3 (7 punti)
Si consideri il linguaggio D di Dyck con coppie di parentesi a, a' e b, b', cioè quello definito dalla grammatica S → SS | a S a' | b S b' | a a' | b b'.
Si descrivano esaurientemente una MT a *nastro singolo* e una macchina RAM che accettano D, valutandone le complessità temporale e spaziale, a criterio di costo costante e logaritmico.

Esercizio 4 (9 punti)



Si consideri un albero ternario i cui nodi possono essere di tre diversi colori: verdi, marroni e arancioni di cui un esempio è rappresentato in figura.
Vogliamo individuare i sottografi (che siano anche alberi) più grandi in esso contenuti, costituiti unicamente da nodi arancioni. Ogni sottografo è identificato dal nodo in esso più vicino alla radice dell'albero, detto nodo di testa. A titolo di esempio, in figura, sono presenti tre sottografi completamente arancioni, i cui nodi di testa sono evidenziati da un contorno tratteggiato.
Si progetti un algoritmo, corredato delle eventuali strutture dati necessarie, che restituisce la lista dei nodi di testa dei detti sottografi, ordinati in ordine decrescente rispetto al numero di nodi contenuti nei rispettivi sottografi.
Ovviamente la valutazione della qualità dell'algoritmo dipenderà anche e soprattutto dalla complessità temporale ottenuta.

Tracce delle soluzioni

Esercizio 1

a)

$\forall x \forall y$ (*stessaDim*(x,y) $\leftrightarrow$($\forall$i (*elem*(x,i)=$\perp$ $\leftrightarrow$ *elem*(y,i)=$\perp$)))

$\forall x \forall y$ (*uguale*(x,y) $\leftrightarrow$ ($\forall$i (*elem*(x,i)=*elem*(y,i))))

$\forall x \forall y$ (*domina*(x,y) $\leftrightarrow$ *stessaDim*(x,y) $\wedge$ ($\forall$i *elem*(x,i) $\geq$ *elem*(y,i)) $\wedge$ ($\exists$i *elem*(x,i) > *elem*(y,i)))

b)

Pre-condizione: specifichiamo che tutti i punti sono diversi e che hanno tutti la stessa dimensionalità.

$\forall x \forall y (x \in P \wedge y \in P \wedge x \neq y \rightarrow \neg$*uguale*(x,y) $\wedge$ *stessaDim*(x,y))

SKYLINE

Post-condizione: specifichiamo che i punti di S sono esattamente quei punti di P non dominati da nessun altro punto di P.

$\forall x (x \in S \leftrightarrow x \in P \wedge \neg \exists y (y \in P \wedge \neg$*uguale*(x,y) $\wedge$ *domina*(y,x)))

Esercizio 2

Tutti i 3 problemi sono decidibili:

a. Le funzioni calcolate da MT di $\mathcal{M}_1$ e di $\mathcal{M}_2$ sono tutte periodiche di periodo $T_1.T_2$. Quindi è sufficiente eseguire il calcolo di $M_1 \in \mathcal{M}_1$ e di $M_2 \in \mathcal{M}_2$ su tutti i valori x, $0 \leq x \leq T_1.T_2$ per verificarne l'equivalenza.
b. E' sufficiente simulare l'esecuzione della MT data sulla stringa in ingresso: siccome esiste solo un numero finito di configurazioni possibili che occupino al massimo n^2 celle di memoria, la simulazione potrà in un numero finito di mosse o i) fermarsi accettando la stringa, o ii) fermarsi rifiutando la stringa o iii) ritrovarsi in un'identica configurazione già incontrata in precedenza: in tal caso la macchina sarebbe costretta ad "andare in loop" e quindi si può concludere che la stringa va rigettata.
c. In questo caso sono possibili più computazioni per ogni singola stringa ma quelle che occupano non più di n^2 celle di memoria (e non vanno in loop) sono comunque finite: è quindi sufficiente enumerarle tutte e verificare se almeno una di esse accetta la stringa in ingresso.

Esercizio 3

MT: La macchina procede a passate successive sulla stringa, cancellando ogni coppia a a' o b b' che trova, se non ne trova si blocca. La macchina accetta se riesce a cancellare tutti i caratteri della stringa di ingresso.
Complessità: una passata dell'ingresso di lunghezza n avviene in $\Theta(n)$. Ad ogni scansione viene cancellata (caso pessimo) una sola coppia, riempiendo il "buco" che lascia. Quindi, se la MT accetta, essa farà al più n/2 passate: $T(n) = \Theta(n^2)$. $S(n) = \Theta(n)$, non usando la macchina altra memoria.

RAM: posso simulare il funzionamento dell'automa a pila deterministico naturale per D, con complessità a costo costante $T(n) = \Theta(n)$ e $S(n) = \Theta(n)$. Per il costo logaritmico la differenza è dovuta al costo di indirizzamento dei dati di pila in memoria; risulta dunque $T(n) = \Theta(n \log n)$ e $S(n) = \Theta(n)$.

Esercizio 4

Un procedimento che raggiunge il limite inferiore di complessità (Omega(n): è necessario almeno esaminare tutti i nodi dell'albero una volta) è il seguente:
– Si inizializza una lista di strutture contenenti un riferimento ad un nodo dell'albero e un intero. Il ruolo della lista è quello di contenere i riferimenti ai nodi di testa dei sottografi e il numero di nodi in essi presenti.
– Si aggiunge ad ogni nodo dell'albero un riferimento ad un elemento della lista. Lo scopo del riferimento è tener traccia di quale sia il sottografo costituito unicamente da nodi arancioni a cui il nodo appartiene. I riferimenti sono inizializzati tutti a NIL

Si effettua dunque una visita in preordine dei nodi dell'albero ternario, nella quale si identificano tutti i sottoalberi. In particolare:

- se un nodo visitato è marrone o verde, non occorre fare nulla.
- Se un nodo *n* visitato è arancione, e il padre non lo è, si aggiunge in testa alla lista che tiene traccia dei nodi di testa il riferimento ad esso. Si imposta la dimensione del sottografo, contenuta nell'elemento appena aggiunto alla lista, a 1. Si imposta infine il campo del riferimento al nodo di testa del nodo arancione del sottografo in esame all'elemento appena aggiunto nella lista
- Se un nodo visitato è arancione e il padre è a sua volta arancione, si imposta il riferimento del figlio al nodo della lista che contiene il nodo di testa del sottografo a cui esso appartiene allo stesso valore del riferimento alla lista del padre, e, utilizzando il riferimento appena assegnato, si incrementa il contatore dell'elemento della lista

Terminata la visita in preordine (complessità Theta(n)), si possono ordinare i nodi di testa contenuti nella lista utilizzando un counting-sort, dato che il range di valori ammissibili per le dimensioni dei sottoalberi è $\{0,\ldots,n\}$.
La complessità totale è dunque Theta(n).