Algoritmi e Principi dell'Informatica

Appello del 25 Giugno 2013

Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 3 ore.

Chi deve sostenere solo il modulo di *Informatica teorica* deve svolgere gli Esercizi 1 e 2, in 1 ora e 30 minuti.

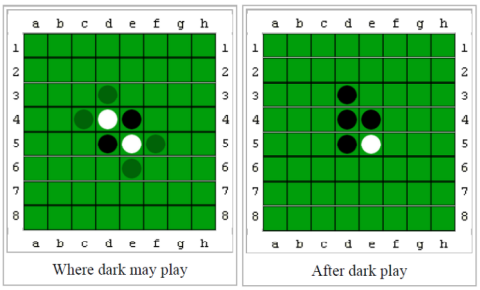
Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 3, 4, 5 in 1 ora e 30 minuti.

NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

Esercizio 1 (punti 7)

Nel gioco *Reversi* ci sono due giocatori ("bianco" e "nero"), ciascuno dotato di 32 dischi bicolori (neri da un lato, bianchi dall'altro) da piazzare su una scacchiera di 8x8 caselle inizialmente vuota.

Si muove alternativamente (inizia il nero) piazzando ogni volta un nuovo disco (girato dalla parte del proprio colore) in una casella vuota. Se tra il disco appena piazzato e quelli del proprio colore già presenti sulla scacchiera sono presenti orizzontalmente, verticalmente o diagonalmente solo dischi avversari senza caselle vuote nel mezzo, tali dischi vengono rovesciati e diventano del colore di chi ha eseguito la mossa. La figura sottostante mostra un esempio di mossa del nero che produce il rovesciamento di un disco bianco.



Per semplicità rispetto alla versione ufficiale del gioco, si assuma che sia sempre possibile muovere, ossia piazzare nuovi dischi, fino a che la scacchiera non sia piena. A fine partita, vince il giocatore con il maggior numero di dischi del proprio colore.

Scopo dell'esercizio è specificare *alcune* (non tutte!) regole del gioco mediante formule del prim'ordine.

Si considerino i predicati seguenti, che fanno riferimento al tempo (discreto) in cui avviene una mossa (ossia al numero d'ordine della mossa all'interno della partita, partendo da 1):

scacchiera(X,Y,C,T) // c'è un disco di colore C al tempo T in posizione (X,Y)

mossa(X,Y,C,T) // il giocatore di colore C piazza un disco del suo colore al tempo T in posizione (X,Y)

1. Si specifichino le condizioni seguenti:
    a. inizialmente (al tempo 0) la scacchiera è vuota e la prima mossa viene eseguita dal nero al tempo 1.
    b. in ogni casella c'è sempre al massimo un solo disco.

2. Si specifichi ora la seguente regola, supponendo che sia già stato definito un predicato *daRibaltare*(X,Y,T) che indica che un disco in posizione (X,Y) in conseguenza di una mossa effettuata al tempo T deve essere ribaltato nello stesso istante T:
   All'istante T viene eseguita una mossa (se nell'istante precedente esistevano caselle vuote); se la mossa produce l'effetto che qualche casella sia ribaltabile, contestualmente, ossia *nello stesso istante* T, avvengono tutti i ribaltamenti possibili.
   NB: *non* è necessario formalizzare quale sia il giocatore che effettua la mossa.

3. [**Opzionale**: questa parte, se risolta in maniera corretta comporta **3 punti** *aggiuntivi*; tuttavia essa *verrà valuatata solo se le parti precedenti dell'esercizio saranno state risolte correttamente*].
   Specificare il predicato *daRibaltare*(X,Y,T) (Suggerimento: si prenda in considerazione questa proprietà a partire dall'istante 1) . In tal caso si consideri inoltre già specificata la seguente definizione di avversario mediante la funzione *avv* (avversario): *avv*(*bianco*) = *nero; avv*(*nero*) = *bianco*

NB: Si ribadisce che *non* è richiesta la specifica di ulteriori regole del gioco (e.g.: una sola mossa per volta, alternanza tra giocatori, …).

Esercizio 2 (punti 8)

Si risponda ai seguenti quesiti, giustificando *brevemente ma con chiarezza* la risposta:

1. Si consideri l'insieme $S = \{ y \mid \forall x (f_y(x)\neq\perp\rightarrow f_y(x)=x^2) \}$. *S* è ricorsivo? S è ricorsivamente enumerabile?
2. Si consideri la funzione seguente:

   $g(x, y) = 1$, se $\forall z (f_x(z)\neq\perp\wedge f_y(z)\neq\perp\rightarrow f_x(z)= f_y(z))$, 0 altrimenti

   g è computabile?
3. Si consideri ora la funzione *g*', variazione della precedente *g*:

   $g'(x, y, z) = 1$, se $f_x(z)\neq\perp\wedge f_y(z)\neq\perp\wedge f_x(z)= f_y(z)$, $\perp$ altrimenti

   g' è computabile?

Esercizio 3 (punti 7)

Si consideri il linguaggio

$L = \{b^{n_1}a^{m_1}b^{n_2}a^{m_2}\ldots b^{n_h}a^{m_h}|$

(1) $k>0$, (2) $n_i\geq n_{i+1}$ $i=1, \ldots, k-1$, (3) $1\leq n_i\leq 10$, $i=1,\ldots, h$

(4) $m_{i+1}\geq m_i$ $i=1, \ldots, k-1$, (5) $1\leq m_i\leq 5$, $i=1,\ldots, h\}$.

1) Quale è la complessità di una MT a k nastri che riconosce L?

2) Che cosa cambierebbe con una MT a nastro singolo?

3) È possibile ottenere la stessa complessità con una RAM usando il criterio di costo logaritmico?

4) Le risposte ai punti precedenti cambierebbero se venisse rimosso il vincolo (5)? Se sì, come?

Per tutti i punti, si descriva anche solo schematicamente il comportamento della MT o della RAM usata per risolvere il problema, in modo da motivare la complessità fornita.

Esercizio 4 (punti 5)

a. Dato un albero binario di interi, memorizzato mediante una struttura dati che prevede che *ogni nodo punti ai propri figli ma non al padre* e che può contenere elementi ripetuti, si vogliono cancellare tutte le foglie che contengono un valore uguale a quello del padre. Si fornisca una breve traccia di un algoritmo per questo scopo e se ne valuti la complessità asintotica; essa è ottima?
   NB: non è richiesto il codice e neanche lo pseudocodice dell'algoritmo: è sufficiente una sintetica ma precisa descrizione che permetta di valutarne la complessità in modo evidente.
b. Si voglia estendere l'algoritmo in modo tale da ottenere un albero in cui non ci siano foglie che contengono il valore del padre. Fornire una traccia e la relativa complessità del nuovo algoritmo.

Esercizio 5 (punti 5)

Si consideri il seguente problema (versione semplificata di un classico problema per la classificazione di dati):

Dato un insieme N di punti in uno spazio m-dimensionale partizionarli in un numero k – assegnato e minore di N – di *cluster* (sottoinsiemi disgiunti) così che ogni punto appartenga ad un solo cluster, ed ogni cluster contenga almeno un punto. Ogni cluster ha un *centroide* definito come la media, nello spazio euclideo m-dimensionale, dei punti che appartengono al cluster stesso (ad esempio, se il cluster è costituito da due punti, il suo centroide è il punto di mezzo tra essi).

Un semplice metodo greedy (che non garantisce di trovare la migliore separazione possibile fra i cluster, che nella presente versione semplificata non è defintia) consiste nello scegliere casualmente k punti come centroidi iniziali dei k clusters. Quindi ogni altro punto dell'insieme viene assegnato al cluster il cui centroide è il più vicino ad esso e subito dopo viene ricalcolato il centroide di quel cluster per tenere conto del nuovo valore; si procede in tal modo fino ad esaurimento dei punti.

Si dettagli la precedente traccia di algoritmo mediante opportuno pseudocodice fino a un livello di dattaglio tale da poterne definire con precisione la complessità asintotica (temporale e spaziale).

Informazione di completamento culturale, irrilevante allo scopo di risolvere l'esercizio: l'algoritmo completo procede per iterazioni successive fino a quando il calcolo dei centroidi e la clusterizzazione non raggiungono un opportuno criterio di ottimalità.

Tracce delle soluzioni

Esercizio 1

1.
   a. $\forall X\ \forall Y\ \forall C\ \neg$*scacchiera*$(X,Y,C,0)\wedge\neg$*mossa*$(X,Y,C,0)$
      $\wedge\exists\ Z,W$ *mossa*$(Z,W,$ *nero*$,1)$
   b. $\forall X\ \forall Y\ \forall T\ \neg($*scacchiera*$(X,Y,$*bianco*$,T)\wedge$*scacchiera*$(X,Y,$*nero*$,T))$

2.

$\forall T\ (T > 0 \wedge \exists X,Y(\neg\exists C$ *scacchiera*$(X,Y,C,T\text{-}1))\rightarrow$

$\exists Z,W,H$ *mossa*$(Z,W,H,T)\wedge$

$(\neg\exists K$ *scacchiera*$(Z,W,K,T\text{-}1))\wedge$

scacchiera(Z,W,H,T)

$\wedge$

$(\forall Z\ \forall W$ *daRibaltare* $(Z,W,T)\rightarrow$*scacchiera*$(Z,W,H,T))$

)

//se c'è almeno una casella libera a T-1, a T si fa una mossa che consiste nel mettere un disco in una della caselle libere (a T-1); di conseguenza tutte le caselle divenute ribaltabili sono ribaltate.

Esercizio opzionale

$\forall X\ \forall Y\forall T$ *daRibaltare* $(X,Y,T)\leftrightarrow$

$(T >0\ \wedge$

$(\exists Z,W,P$ *mossa*$(Z,W,P,T)\wedge\exists N\ (N>0)\wedge\exists M\ (M>0)\wedge$

// 1° caso: orizzontale: X e Y si trovano sulla stessa linea orizzontale del punto di una mossa e internamente a un "segmento" che parte dal punto della mossa e termina nel primo punto di uguale colore della mossa(alla sua destra); inoltre tutti i punti interni del segmento, inclusi X e Y, sono di colore opposto.

scacchiera$(Z+M,W,P,T\text{-}1)\wedge$

$Z<X\ \wedge\ X<Z+M\ \wedge\ W = Y\wedge$

$(\forall I(Z < I\ \wedge\ I< Z+M)\rightarrow$*scacchiera*$(I,Y,$*avv*$(P),T\text{-}1))$

$\vee$

//analogamente il caso in cui il segmento parta dal punto di uguale colore della mossa (alla sua sinistra) e termini nel punto della mossa

….

// analogamente per i casi verticali e diagonali

….

))

Formulazione alternativa

$\forall X\ \forall Y \forall T$ ribaltabile(X,Y,T) $\leftrightarrow$

$T>0 \wedge \exists N\ N>0 \wedge \exists M\ M>0 \wedge \exists P$

(

// 1° caso: orizzontale

(mossa(X-N,Y,P,T) $\wedge$ scacchiera(X+M,Y,P,T-1)

$\vee$

mossa(X+M,Y,P,T) $\wedge$ scacchiera(X-N,Y,P,T-1))

$\wedge$

$\forall I$ (X-N<I$\wedge$I<X+M $\rightarrow$scacchiera(I,Y,avv(P),T-1)))

$\vee$

// analogamente per i casi verticali e diagonali

…

)

Esercizio 2

1. *S* non è ricorsivo: ciò è una diretta conseguenza del toerema di Rice.D'altro canto il complemento di *S*, *S'*:

   $S' = \{ y \mid \exists x (f_y(x) \neq \perp \wedge f_y(x) \neq x^2) \}$

   è ricorsivamente enumerabile: i suoi elementi possono essere enumerati mediante una classica tecnica "diagonale" (su 3 dimensioni) che consideri nell'ordine tutte le funzioni computabili, per tutti i valori dei dati di ingresso e per numero crescente di passi di computazione individuando quindi i valori di *y* per i quali la condizione $f_y(x) \neq x^2$ sussiste. Di conseguenza *S* non è ricorsiviamente enumerabile altrimenti *S* e *S'* sarebbero entrambi ricorsivi.
2. Sia K il numero di Godel di una MT che calcola la funzione $f(x)=x^2$: risulta allora $g(x, K)=p(x)$, dove p è la funzione caratteristica dell'insieme S di cui al punto 1; quindi $g(x, K)$ non è computabile, ed a maggior ragione non lo è $g(x,y)$.
3. *g'* è computabile: una MT può calcolare *g'* simulando il calcolo di $f_x(z)$ e $f_y(z)$ e, se entrambe le computazioni terminano, verifica se $f_x(z)= f_y(z)$, altrimenti continua con la sua simulazione indefintamente.

Esercizio 3

Il linguaggio L è regolare, grazie ai vincoli (3) e (5); quindi, potendo ognuna delle macchine prese in considerazione simulare il funzionamento di un automa a stati finiti con il proprio organo di controllo, ognuna di esse può riconoscere L con complessità temporale O(n) e spaziale O(1), con l'eccezione della macchina di Turing a nastro singolo che deve usare n celle per la stringa di ingresso.

Se si rimuove il vincolo (5) L non è più regolare (e neanche context-free). Una MT a 2 nastri può memorizzare i vari m_i su un nastro e m_{i+1} sull'altro in alternanza, contestualmente verificando che sia soddisfatta la diseguaglianza; ciò richiede evidentemente complessità spaziale e temporale O(n). Una macchina RAM può procedere in maniera analoga usando 2 contatori, ottenendo una complessità temporale O(n log n) col criterio di costo logaritmico. Una macchina a nastro singolo invece deve scandire ogni gruppo di m_i 'a' m_i volte, ciò che comporta una complessità temporale $O(n^2)$.

Esercizio 4

a. E' sufficiente una visita in preordine dell'albero. Quando si raggiunge una foglia si verifica se il padre ha lo stesso valore e in tal caso la si elimina. Quindi la complessità è lineare nel numero dei nodi.
b. La soluzione più naturale consiste nell' iterare l'algoritmo precedente. Ogni iterazione ha quindi complessità lineare nel numero dei nodi; il numero di iterazioni al massimo è dato dalla profondità dell'albero. Quindi nel caso pessimo la complessità è $O(n^2)$. Tuttavia, un algoritmo più sofisticato può tenere traccia, ovviamente in maniera ricorsiva, durante la visita dell'albero, del fatto che un nodo sia diventato foglia a seguito della cancellazione dei suoi figli: in tal caso esso viene cancellato e, al momento di ritornare il controllo al proprio padre gli notifica il fatto di essere stato cancellato; se un nodo riceve questa ionformazione da ambo i figli sa di essere a sua volta diventato foglia e il meccanismo può essere iterato durante la seuqnze di ritorno alle varie chiamate ricorsive, ottenendo quindi una complessità O(n).

Esercizio 5

Complessità temporale:

considerando costante il tempo per calcolare la distanza del punto dai k centroidi (in realtà dipenda da m ma non da k né da N), ognuna delle N iterazioni richiede k confronti ed il calcolo del nuovo centroide, quindi O(N) nel caso pessimo dal momento che tutti i punti potrebbero essere assegnati ad un unico cluster. La complessità temporale totale è perciò $O(N^2).m$. Si noti tuttavia che il ricalcolo della media dei punti (centroide) a causa dell'inserimento del nuovo punto potrebbe essere fatto anche in O(1) moltiplicando la media precedente per il numero *s* di punti precedente; aggiungendo le coordinate del nuovo punto e ridividendo per *s+1*. In tal modo la complessità diventerebbe O(N.k.m)

Complessità spaziale

occorre tenere in memoria i punti ed i centroidi, quindi O((N + k)m)