Informatica Teorica (Sez. Cremona+Como)

Appello del 6 Febbraio 2006

Esercizio 1 (Punti 15)

Si formalizzino mediante formule logiche del prim'ordine le *regole* del gioco "Sudoku".
Informalmente le regole sono le seguenti:
E' dato un quadrato di 9 * 9 caselle, ognuna delle quali deve contenere un numero intero compreso tra 1 e 9. Il quadrato è suddiviso in 9 "sottoquadrati" di 3 * 3 caselle (il primo "copre" le caselle [1..3 * 1..3]; il secondo [4..6* 1..3], ecc.)
Occorre riempire le caselle del quadrato in modo che:

- Ogni riga e ogni colonna contenga tutti i numeri tra 1 e 9
- Ogni "sottoquadrato" contenga tutti i numeri tra 1 e 9.

Facoltativamente (ulteriori punti 1): potrebbe il problema essere generalizzato rispetto alle dimensioni del quadrato? Se sì, come?

Esercizio 2 (punti 7)

Si dica, giustificando brevemente la risposta, se il suddetto problema del Sudoku, ossia stabilire se, assegnati alcuni valori ad alcune caselle esiste un modo di riempire le altre in modo da soddisfare le regole, è decidibile o no.
Come cambia la risposta (se cambia), se il problema è espresso nella sua forma generalizzata?

Esercizio 3 (punti 8)

Si descrivano brevemente (senza necessariamente "codificarli" in dettaglio) algoritmi per il riconoscimento del linguaggio $\{a^n b^n \mid n >= 1\}$ con le seguenti caratteristiche:

- Una macchina di Turing che minimizzi la complessità temporale
- Una macchina di Turing che minimizzi la complessità spaziale
- Una RAM che minimizzi la complessità temporale
- Una RAM che minimizzi la complessità spaziale

E si confrontino le funzioni di complessità così ottenute.

Soluzioni

Esercizio 1

Si formalizzi la tabella del Sudoku mediante un array (si ricordi che, in termini matematici, un array altro non è che una funzione). Le regole del gioco sono allora formalizzate dalla formula seguente:

$$\forall i,j(1 \le i,j \le 9 \rightarrow 1 \le a[i,j] \le 9)$$
$$\wedge$$
$$\forall i,j,k(1 \le i,j,k \le 9 \rightarrow a[i,j] \ne a[i,k]$$
$$\wedge$$
$$\forall i,j,k(1 \le i,j,k \le 9 \rightarrow a[i,j] \ne a[k,j]$$
$$\wedge$$
$$\forall i,j,k,h((1 \le i,j,k,h \le 9 \wedge i/3 = k/3 \wedge j/3 = h/3 \wedge (i \ne k \vee j \ne h)) \rightarrow a[i,j] \ne a[k,h]$$

NB. il simbolo '/' indica la divisione a risultato intero.

La generalizzazione al caso di un quadrato k*k è banale se si ha l'accortezza di scegliere un valore di k che sia un quadrato perfetto. In tal caso k rimpiazza il numero 9 e $\sqrt{k}$ rimpiazza il numero 3 nella formula precedente.

Esercizio 2

Il problema è chiaramente decidibile in quanto è riconducibile del tutto a un problema finito. Infatti, ogni griglia 9 * 9 può essere riempita con numeri da 1 a 9 (rispettando le regole) in un numero finito di modi diversi. (Contare il numero esatto di schemi diversi non è banale, ma è sufficiente capire che esso è finito. Per curiosità, tale numero è stato calcolato essere uguale a 6 670 903 752 021 072 936 960). Pertanto un algoritmo che provi esaustivamente tutte le possibili soluzioni termina sicuramente.

Anche nal caso generalizzato, il problema rimane decidibile. Per ogni *n* fissato, ogni schema di n^2 * n^2 celle ha un numero finito di possibili completamenti (sicuramente meno di $n^{2n\text{^}4}$), per cui possono essere provati esaustivamente tutti.
Il problema è però computazionalmente oneroso, NP-completo per la precisione, come mostrato in: T. Yato, "Complexity and completeness of finding another solution and its application to puzzles". Master's thesis, University of Tokyo, Department of Information Sciences, 2003.

Esercizio 3

1. Macchina di Turing che minimizzi la complessità temporale.
   E' sufficiente "simulare" con la macchina di Turing un automa a pila, usando un nastro di memoria come una pila. La complessità temporale risultante sarebbe $\Theta(n)$, che è facile capire essere il meglio ottenibile.
2. Macchina di Turing che minimizzi la complessità spaziale.
   In questo caso conviene contare in codifica binaria (o equivalentemente, in altra base > 1) il numero di 'a' ricevute e poi decrementare il contatore binario per ogni 'b' ricevuta. La complessità spaziale sarebbe così $\Theta(\log n)$, ottenuta però al prezzo di un peggioramento di un fattore logaritmico della complessità temporale.
3. RAM che minimizzi la complessità temporale.
   Si può replicare la soluzione della macchina di Turing al punto 1 con una RAM, scrivendo un marker in n celle adiacenti. La complessità temporale risultante è la stessa $\Theta(n)$, con criterio di costo costante. Con criterio di costo logaritmico essa sale invece a $\Theta(n \log n)$, dal momento che anche se scrivo un valore costante (il marker) nelle celle adiacenti, ad ogni accesso devo manipolare un puntatore all'elemento corrente, che è un numero maggiorato da n.
4. RAM che minimizzi la complessità spaziale.
   In questo caso conviene utilizzare un singolo contatore intero in una singola cella di memoria. In questo modo il costo a criterio di costo costante è semplicemente $\Theta(1)$ (uso di un numero costante di celle di memoria), mentre a costo logaritmico diventa $\Theta(\log n)$, che è una caratterizzazione del fatto che la RAM di fatto "implementa" una codifica binaria del dato.