Informatica Teorica - II prova in itinere - 29 Giugno 2005, Sezione Pradella

AVVISO: *Si ricorda che possono svolgere questa prova in itinere solo coloro che abbiano ottenuto almeno 5 punti nella prima prova.*

Eccezionalmente sono ammessi anche gli studenti laureandi, ***che devono indicare esplicitamente la loro situazione nella testata del loro elaborato****. Come già comunicato, in caso di esito positivo nella presente prova, verranno loro comunicate le modalità di recupero della prima prova.*

Tuttavia, i laureandi –con meno di 5 punti nella prima prova- che non dovessero superare l'esame attraverso questa procedura eccezionale ***non potranno sostenere il recupero del 13*** *Luglio.*

Esercizio 1 (punti 6/17-esimi)

Si *specifichi*, mediante opportune pre- e post-condizioni il seguente requisito per un frammento di programma FP:

Sia dato un array *a* di $n < NMAX$ interi positivi; gli *n* interi sono memorizzati nelle prime posizioni di *a*, a partire da *a*[0]; dopo di essi si trova il valore convenzionale 0 usato come "terminatore".

FP deve produrre un nuovo array *b*, che sia il risultato dell'ordinamento di *a* in ordine crescente e dell'eliminazione di eventuali ripetizioni. FP usa *a* come variabile "read-only". Anche la sequenza dei valori di *b* deve essere terminata da uno 0.

Suggerimento: si consiglia di introdurre la funzione *lungh(z)* definita su array di NMAX interi che fornisce il numero di valori di *z* diversi da 0 che precedono il primo 0 (in caso di assenza di uno 0, *lungh(z)* = -1 per convenzione).

Esercizio 2 (punti 7/17-esimi)

Si consideri il programma seguente, codificato in un ipotetico linguaggio ispirato al C.

```
int g(int p); int f(char t);
main ()
{  char a[20]; int i, z;
   scanf(a); z = 0;
   while (i = 0; g(i) != 20; i++) do z = z + f(a[i]);
   printf(z);
}
```

Si assuma che f e g siano due sottoprogrammi "esterni" la cui terminazione sia garantita per ogni valore dei parametri di ingresso.

Si dica, motivando brevemente la risposta, se è decidibile il problema di stabilire se il programma main di cui sopra terminerà o meno la sua esecuzione in corrispondenza di un generico –non fissato!- valore del dato di input a.

Esercizio 3 (punti 7/17-esimi)

Si consideri una sequenza S di elementi che contengano informazione codificabile in un numero limitato a priori di celle di memoria. Si assuma che S si trovi già memorizzata nella memoria di una macchina RAM secondo una delle due tradizionali tecniche:

a) come array di celle consecutive

b) come lista di elementi collegati tra loro mediante puntatori

Si valuti la complessità asintotica sia spaziale che temporale, a meno della relazione Θ, di un algoritmo di ricerca sequenziale nei due casi, facendo uso sia del criterio di costo costante che del criterio di costo logaritmico; si spieghi brevemente come si giunge ai risultati proposti, senza bisogno peraltro di codificare in dettaglio l'algoritmo. Si indichi con precisione il parametro rispetto al quale viene misurata la dimensione dei dati di ingresso, in funzione della quale viene fornita la funzione di complessità.

Esercizio 3-bis, Facoltativo (punti 4/17-esimi, valido solo se preceduto da soluzione corretta della parte obbligatoria)

Come cambiano le valutazioni di complessità di cui sopra immaginando che il medesimo algoritmo venga eseguito da una Macchina di Turing?

Soluzioni

Esercizio1

In primo luogo si definisca la funzione *lungh(z)* mediante una formula del tipo

$$lungh(z) = n \leftrightarrow ((0 \le n < NMAX \wedge z[n]=0 \wedge \neg\exists i(0 \le i < n \wedge z[i] = 0)) \vee (n = -1 \wedge \neg\exists i(0 \le i < NMAX \wedge z[i] = 0)))$$

A questo punto il requisito richiesto può essere specificato dalla seguente coppia pre-post-condizione:

$$\{\exists n\,(lungh\,(a) = n \quad \wedge \forall i\,(0 \le i < n \rightarrow a[i] > 0))\}$$
$$FP$$
$$\{\exists m\,(lungh\,(b) = m \quad \wedge$$
$$\forall i\,(0 \le i < m-1 \rightarrow b[i] < b[i+1]) \quad \wedge$$
$$\forall i\,(0 \le i < lungh\,(a) \rightarrow \exists j\,((0 \le j < m) \wedge (a[i] = b[j]))) \quad \wedge$$
$$\forall j\,(0 \le j < m \rightarrow \exists i\,((0 \le i < lungh\,(a)) \wedge (a[i] = b[j])))\}$$

Esercizio2

La risposta è positiva

Premessa. Perché il quesito sia significativo occorre ovviamente assumere una macchina astratta che presupponga un dominio totale di dati infinito: in questo caso, il tipo **int** deve essere l'insieme matematico degli interi. Altrimenti, come noto, la macchina astratta diventa automaticamente un automa a stati finiti e quindi la sua terminazione decidibile.

Il ***dominio*** dei dati di ingresso *del programma in oggetto* è però ***finito***, essendo l'insieme degli array di 20 caratteri. Per ognuno dei possibili valori di ingresso la risposta al quesito è semplicemente un sì o un no; esiste quindi una macchina di Turing che fornisce tale risposta, per ogni valore del dato di ingresso a; di conseguenza esiste un numero finito di macchine di Turing, che, opportunamente combinate tra loro, forniscono la risposta corretta per ogni valore del dato di ingresso a; la combinazione di tali macchine può dar luogo ad un'unica macchina che, sul dominio finito dei possibili valori di a, fornisce l'insieme finito delle risposte corrette. Ovviamente, ciò non garantisce la conoscenza di tale macchina; ne garantisce esclusivamente l'esistenza.

Equivalentemente e più sinteticamente si può osservare che il problema posto (la decidibilità dell'halt del programma rispetto ai diversi valori di ingresso) è formalizzato da una funzione a dominio finito (l'insieme degli array di 20 caratteri) e codominio {T, F}. Tale funzione è quindi riducibile a una tabella finita, e quindi calcolabile.

Si noti anche che la terminazione del programma, essendo garantita la terminazione di una singola esecuzione del corpo del ciclo, dipende solo dalla valutazione della funzione g, che, o non produce mai 20, oppure prima o poi deve produrre il valore 20 per qualche valore di i. Ciò è indipendente dal valore del dato di ingresso a. Quindi la tabella di cui sopra, non dipendendo dai dati di ingresso sarà costituita da tutti T o tutti F.

Esercizio 3

Si indichi con n il numero di elementi della sequenza. Assumiamo per semplicità che ogni elemento contenga informazione codificabile in un numero fissato di byte. In questo caso è del tutto indifferente assumere come parametro di dimensione dei dati in ingresso il numero n di elementi di S o le celle di memoria utilizzate per memorizzare S.

La memorizzazione mediante array richiede una quantità di memoria $\Theta(n)$ sia a criterio di costo costante che a criterio logaritmico. La memorizzazione mediante puntatori invece richiede di memorizzare un intero come puntatore per ogni elemento di S. Essendo il numero di elementi di S illimitato, una quantità di memoria richiesta sarà $\Theta(n)$ a criterio di costo costante ma $\Theta(n \cdot \log(n))$ a criterio logaritmico.

La complessità temporale risulterà invece $\Theta(n)$ a criterio di costo costante e $\Theta(n \cdot \log(n))$ a criterio logaritmico per entrambi i tipi di memorizzazione.

Esercizio 3 bis (parte facoltativa)

Simulare un array mediante un nastro di macchina di Turing richiede ovviamente uno spazio $\Theta(n)$. $\Theta(n)$ risulta pure la complessità temporale di un algoritmo di ricerca sequenziale.

La memorizzazione di un puntatore in un nastro di macchina di Turing richiede la codifica (ad esempio in binario) di un numero i e quindi un numero $\Theta(\log(i))$ di celle. La complessità spaziale risulta perciò $\Theta(n \cdot \log(n))$ e l'accesso a un generico elemento richiede pure $\Theta(n \cdot \log(n))$, rendendo perciò la complessità totale $\Theta(n^2 \cdot \log(n))$