Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

4-9-2024

Informatica teorica

Esercizio 1 (8 punti)
Si consideri la seguente formula F:

$$\forall x \forall y \forall z \left((a(x) \wedge a(y) \wedge a(z) \wedge x < y \wedge y < z) \rightarrow (\exists x'(x < x' \wedge x' < y) \vee \exists y'(y < y' \wedge y' < z)) \right)$$

1. Si descriva il linguaggio $L' = L(F)$ assumendo come alfabeto $\Sigma = \{a\}$.
2. Si descriva il linguaggio $L'' = L(F)$ assumendo come alfabeto $\Sigma = \{a, b\}$.
3. Si costruiscano due automi o grammatiche a potenza minima per L' e L''.

Soluzione

1. $L' = \{w \in \Sigma^* \mid |w| < 3\}$.
2. $L'' = \neg(\Sigma^* a^3 \Sigma^*)$.
3. L':
   $S \rightarrow \varepsilon \mid aA \mid a$
   $A \rightarrow a$

   L'': $S \rightarrow \varepsilon \mid bS \mid b \mid aA \mid a$
   $A \rightarrow bS \mid b \mid aB \mid a$
   $B \rightarrow bS \mid b$.

Esercizio 2 (8 punti)

Data una funzione $f(x)$, il suo dominio D_f è dato dall'insieme di tutti quei valori in input tali che il risultato è definito: $D_f = \{x \mid f(x) \neq \perp\}$.

Similmente, il codominio (range) di f, R_f, è dato dall'insieme di quei valori y tali per cui esiste almeno un input x per cui vale $f(x) = y$, ovvero $R_f = \{y \mid \exists x f(x) = y\}$.

1. È decidibile il problema di stabilire se, date due generiche macchine di Turing M_x e M_y, le corrispondenti funzioni da loro calcolate f_x ed f_y sono tali che il dominio di f_x è un sottoinsieme del dominio di f_y?
2. È decidibile il problema di stabilire se, date due generiche macchine di Turing M_x e M_y, le corrispondenti funzioni da loro calcolate f_x ed f_y sono tali che il codominio di f_x è un sottoinsieme del codominio di f_y?

Soluzione

1. Il problema non è decidibile. Si fissi M_x considerando, per esempio, una macchina di Turing che calcola la funzione costante uguale ad 1 (indichiamo con $\bar{x}$ il suo numero di Gödel). Le macchine di Turing M_y tali che il dominio di $M_{\bar{x}}$ è incluso nel dominio di M_y sono tutte e sole quelle che calcolano funzioni totali. Quindi, se riuscissimo a decidere il problema originario (più generale), potremmo sfruttarlo per decidere il problema della totalità di una funzione calcolata da una macchina di Turing, che è notoriamente indecidibile.

2. Anche questo problema non è decidibile. Questo si può mostrare in modo molto simile al punto 1. Più precisamente, si fissi in questo caso M_y considerando, per esempio, la macchina di Turing che calcola la funzione indefinita ovunque (indichiamo con $\bar{y}$ il suo numero di Gödel). Le macchine di Turing M_x tali che il codominio di M_x è incluso nel codominio di $M_{\bar{y}}$ sono tutte e sole quelle che calcolano la funzione indefinita ovunque. Quindi, se riuscissimo a decidere il problema originario (più generale), potremmo sfruttarlo per decidere il problema se una macchina di Turing calcola la funzione indefinita ovunque o no, che si dimostra essere indecidibile applicando direttamente il teorema di Rice.

Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

4-9-2024

Algoritmi e strutture dati

Esercizio 3 (8 punti)

Si consideri il problema di ordinare una sequenza di numeri, espressi in notazione binaria naturale, sull'alfabeto $A = \{0, 1, \diamond\}$, dove $\diamond$ è una lettera utilizzata come separatore tra le codifiche di due numeri consecutivi sul nastro di input. Si ricorda che, in notazione binaria naturale, le cifre binarie compiaiono, da sinistra a destra, dalla più alla meno significativa. Ad esempio, un input costituito dalla sequenza $(1, 2, 3)$ è codificato come $1 \diamond 10 \diamond 11$.

1. Si descriva un algoritmo per macchina di Turing a k nastri, scegliendo un opportuno valore di k, che emette la sequenza di numeri, ordinata in ordine crescente, con la stessa codifica con cui sono ricevuti in ingresso, minimizzando la complessità temporale. Se ne valuti la complessità temporale.

2. Si descriva un algoritmo per macchina RAM che emette la sequenza di numeri, ordinata in ordine crescente, con la stessa codifica con cui sono ricevuti in ingresso, minimizzando la complessità temporale. Se ne valuti la complessità temporale unicamente con il criterio di costo opportuno.

Soluzione

1. È sufficiente una MT a 2 nastri per riuscire ad ottenere la complessità computazionale migliore possibile (ovvero $\mathcal{O}(n^2)$, con n pari alla lunghezza dell'input). Lo schema operativo è il seguente: la MT copia tutto l'input sul primo nastro di memoria, separatori inclusi. A questo punto, la MT adotta l'approccio dell'*insertion sort*. Per tener traccia della porzione già ordinata del nastro di memoria sostituisce il $\diamond$ che segue l'ultimo elemento già ordinato con un blank. Essa memorizza nel secondo nastro di memoria l'elemento da confrontare. Per ogni elemento, la MT deve, nel caso peggiore, scorrere tutto il nastro che lo precede 2 volte, una per localizzare il punto dell'inserimento, un'altra per, materialmente, spostare gli elementi in avanti. La complessità temporale è $\mathcal{O}(n^2)$.

2. La macchina RAM trasferisce nelle celle di memoria i numeri in input, usando una cella per numero. Quest'operazione ha costo lineare nella lunghezza dell'input: è infatti sufficiente inizializzare la cella di memoria che deve contenere un intero a zero, e, per ogni bit letto, in sequenza, raddoppiare il valore contenuto nella cella e sommare il valore del bit letto stesso. La macchina RAM tiene, in una cella dedicata, conto del numero di interi convertiti. L'ordinamento dei numeri in memoria viene effettuato tramite un algoritmo con complessità di caso pessimo ottima, e.g., Heapsort. La macchina RAM stampa quindi in output, convertendo in binario i numeri contenuti in memoria in ordine. Complessità temporale $\mathcal{O}(n \log(n))$ a criterio di costo costante, data dall'ordinamento (le conversioni, letture e stampe hanno costo lineare).

Esercizio 4 (8 punti)

1. Si considerino due array di interi A e B di dimensioni n_A e n_B rispettivamente. Si scriva lo pseudocodice della procedura che determina la distanza $|x - y|$ tra i numeri $x \in A$ e $y \in B$ più vicini tra loro, cioè tali che $|x - y|$ sia il più piccolo possibile. Si fornisca la complessità asintotica in n_A e n_B della procedura.

2. Si voglia ora *massimizzare* $|x - y|$. Cambiano, e in che modo, procedura e complessità?

3. Si voglia ora *minimizzare* $x + y$. Cambiano, e in che modo, procedura e complessità?

Soluzione

1. La soluzione naive che considera tutte le possibili coppie di numeri comporta chiaramente una complessità di $\mathcal{O}(n_A \cdot n_B)$. È possibile però preordinare i due array e scorrerli linearmente mediante degli indici, facendo avanzare solo l'indice dell'elemento più piccolo, in modo tale che il valore assoluto della loro differenza possa diminuire (certamente non può diminuire se si fa avanzare l'indice dell'elemento più grande). Di volta in volta viene calcolata la differenza tra i due valori e si tiene traccia della differenza più piccola. La complessità risultante è quindi pari a $\mathcal{O}(n_A \cdot \log n_A + n_B \cdot \log n_B)$ per ordinare i due array e $\mathcal{O}(n_A + n_B)$ per scorrerli linearmente, quindi complessivamente si ha $\mathcal{O}(n_A \cdot \log n_A + n_B \cdot \log n_B)$.

```
minDifference(A, B)
 1  C := sort(A) // copia ordinata di A
 2  D := sort(B)
 3  i := j := 0
 4  minDiff := +∞
 5  while i < |C| and j < |D|
 6        if |C[i] − D[j]| < minDiff
 7              minDiff := |C[i] − D[j]|
 8        if C[i] < D[j]
 9              i := i + 1
10        else
11              j := j + 1
12  return minDiff
```

2. In questo caso basta una scansione lineare di entrambi gli array per individuarne il minimo e il massimo, con complessità $\mathcal{O}(n_A + n_B)$. La massima differenza in valore assoluto sarà data o dal massimo di A meno il minimo di B o dal massimo di B meno il minimo di A.

3. In questo caso basta una scansione lineare di entrambi gli array per individuarne il minimo, con complessità $\mathcal{O}(n_A + n_B)$.