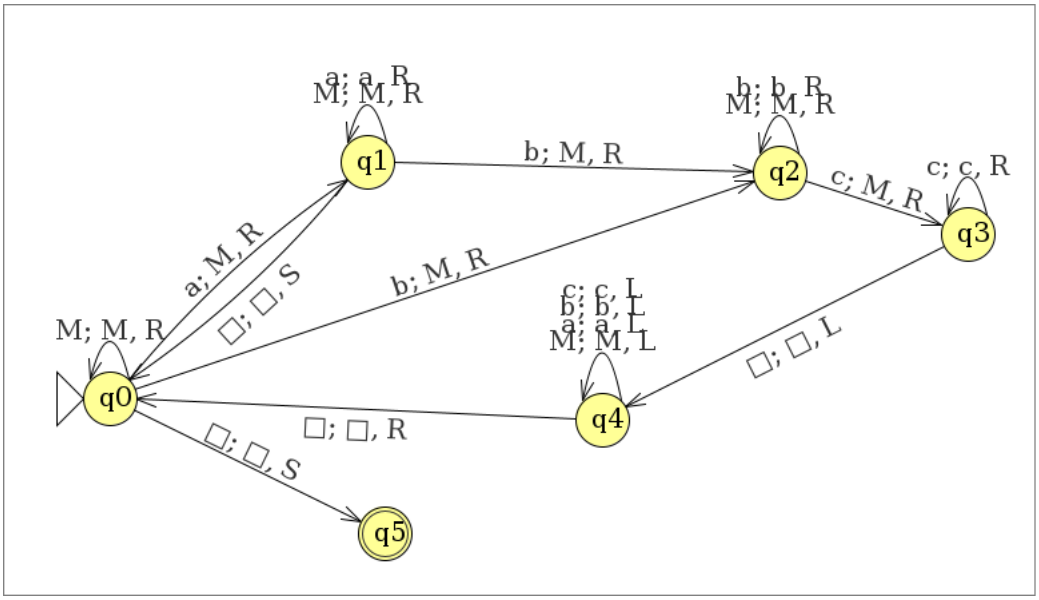
Algoritmi e Principi dell'Informatica – 13-7-2024
Soluzioni al Tema d'esame

Informatica teorica

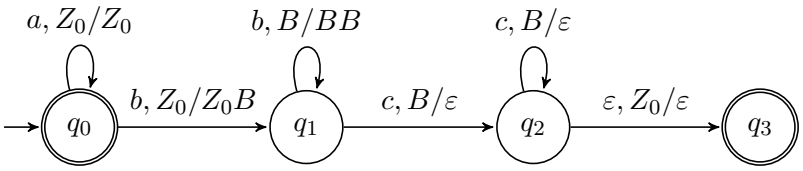
Esercizio 1 (8 punti)
Si consideri il seguente automa M.



1. Si dica di che macchina si tratta.
2. Si descriva esattamente il linguaggio $L(M)$, motivando la risposta.
3. Si scelga un modello a potenza minima per $L(M)$ e si scriva una macchina o grammatica o formula logica corrispondente.

Soluzione

1. Si tratta di una MT a nastro singolo.
2. $L(M) = \{a^* b^n c^n \mid n \geq 0\}$.
3. Basta un automa a pila deterministico:



Esercizio 2 (8 punti)

1) Si consideri la seguente funzione e si dica se essa sia calcolabile o meno, motivando la risposta

$$h(x,y) = \begin{cases} 1 \text{ se } f_{y+1}(x+1) > f_y(x) \\ \perp \text{ altrimenti} \end{cases}$$

2) Si consideri la seguente funzione e si dica se essa sia calcolabile o meno, motivando la risposta

$$h'(x,y) = \begin{cases} 0 \text{ se } f_{y+1}(x+1) \leq f_y(x) \\ \perp \text{ altrimenti} \end{cases}$$

3) Si considerino due generiche MT a nastro singolo, M_i e M_j. Si dica se è decidibile il problema di stabilire se $f_i(i) \leq f_j(j)$. Suggerimento: si può assumere che l'enumerazione $\mathcal{E}$ delle MT inizi con la macchina che si ferma immediatamente e che quindi calcola la funzione $f_0(x) = x$.

Soluzione

1. Calcolabile: basta simulare entrambe le macchine corrispondenti. Se entrambe terminano, si confrontano i due risultati, restituendo 1 in caso positivo; se una delle due non termina, la simulazione non termina come richiesto dalla specifica.

2. Calcolabile, esattamente come il caso precedente. Le uniche differenze sono dovute al diverso confronto e al fatto che si debba restituire 0.

3. Si tratta di capire se la seguente funzione sia calcolabile:

$$t(x, y) = \begin{cases} 1 \text{ se } f_x(x) \leq f_y(y) \\ 0 \text{ altrimenti} \end{cases}$$

   Fissiamo $x = 0$. So che M_0, avendo una funzione di transizione totalmente indefinita, calcola $f_0(x) = x$. Quindi ottengo:

$$t(0, y) = \begin{cases} 1 \text{ se } f_y(y) \geq 0 \\ 0 \text{ altrimenti} \end{cases}$$

   Posso ridurre il problema della terminazione, nel caso diagonale $x = y$, al problema in oggetto. Se $t(0, y)$ fosse calcolabile, $t(0, y) = 1$ determinerebbe la terminazione di $f_y(y)$, mentre $t(0, y) = 0$ la non terminazione.

Algoritmi e Principi dell'Informatica – 13-7-2024
Soluzioni al Tema d'esame

Algoritmi e strutture dati

Esercizio 3 (8 punti)

Si definisce *peso di Hamming* il numero di elementi non nulli di una stringa definita su un alfabeto di cifre. Si consideri l'insieme delle stringhe definite dall'espressione regolare $(0|1)^*.2$ sull'alfabeto di cifre $\Sigma = \{0, 1, 2\}$.

1) Si descriva sinteticamente, ma con sufficiente precisione un algoritmo per macchina di Turing a nastro singolo che calcola il peso di Hamming di una stringa dell'insieme sopracitato, rappresentando il risultato in binario, e se ne valutino complessità temporali e spaziali.

Soluzione

La macchina di Turing scorre il nastro a destra fino ad incontrare la cifra 2 su di esso. Giunta ad essa, scorre verso sinistra fino alla prima cifra 1: se ne incontra una, la sovrascrive con uno 0, torna alla cifra 2 ed incrementa un contatore binario, stoccato dal bit meno significativo al più significativo posto oltre essa. Incrementato il contatore, torna alla cifra 2 e ripete il procedimento di ricerca di 1, sovrascrittura con 0 e incremento fino a quando non sono presenti più 1. Conclude il calcolo riportandosi sulla cifra 2, sovrascrivendola con 0 ed incrementando un'ultima volta il contatore. Complessità temporale $\mathcal{O}(n^2 + n\log(n)) = \mathcal{O}(n^2)$. Complessità spaziale $O(n + \log(n)) = O(n)$.

2) Si descriva una macchina RAM che calcola il peso di Hamming per una stringa dell'insieme sopracitato e stampa il risultato. Se ne valutino complessità temporali e spaziali con l'opportuno criterio di costo.

Soluzione

```
      load= 0
      store 2
      load= 1
      store 3
loop: load 2
      read 1
      add 1
      store 2
      load 1
      sub 3
      jlz loop
      load 2
      sub 3
      write 0
      halt
```

La cella $M[2]$ contiene la variabile intera in cui viene accumulato il peso di Hamming, la cella $M[1]$ contiene la cifra che viene letta di volta in volta dal nastro d'ingresso. L'algoritmo inizializza $M[2]$ a 0, dopodichè legge una cifra, e la somma all'accumulatore. Fatto ciò, controlla se la cifra letta è minore o uguale a 1 sottraendole la costante 1 (la cella $M[3]$ contiene la costante 1). Nel caso questo non accada riprende il ciclo di lettura e somma, altrimenti decrementa di 1 il contatore del peso di Hamming (la cifra non-nulla 2 verrebbe a contribuire per un valore 2 al peso di Hamming), lo stampa e termina. Complessità temporale $\mathcal{O}(n)$ a costo costante, complessità spaziale $\mathcal{O}(1)$.

Esercizio 4 (8 punti)

1) Si scriva lo pseudocodice della procedura che dato un riferimento alla radice di un albero binario di ricerca T, ne "inverte" l'invariante, ovvero restituisce il riferimento alla radice di un albero binario dove tutte le chiavi del sottoalbero destro di un nodo sono minori di quella del nodo, e tutte quelle del sottoalbero sinistro sono maggiori. Il corpo della procedura può allocare al più una quantità costante di variabili scalari. Se ne valuti la complessità temporale.

Soluzione

```
INVERTI(T)
1  tmp = T.left
2  T.left = T.right
3  T.right = tmp
4  INVERTI(T.left)
5  INVERTI(T.right)
6  return T
```

2) Si consideri la famiglia di grafi non orientati $\{G_1, \ldots, G_n\}$ con le seguenti caratteristiche. Ogni nodo di un grafo G_k, con $1 \leq k \leq n$, è etichettato con una delle combinazioni semplici (ovvero senza ripetizioni) di k elementi scelti tra $\{1, \ldots, n\}$ e, per ogni combinazione semplice α di k elementi, c'è un nodo in G_k etichettato con α. Due nodi sono adiacenti se le combinazioni con cui sono etichettati differiscono per un solo elemento.

   1. Si individui, tra le rappresentazioni viste a lezione, quella più efficiente in memoria per questi grafi.
   2. Si riporti la complessità computazionale di determinare se due nodi sono adiacenti.

Soluzione

I grafi in questione hanno $\binom{n}{k}$ nodi, ognuno dei quali ha $k(n-k)$ archi, poiché, per ognuno dei k interi che etichettano il nodo, è possibile scegliere tra $n-k$ interi diversi dai k dell'etichetta. In totale, il numero di archi è quindi $\frac{\binom{n}{k}k(n-k)}{2}$. Questo numero è, per n che tende ad infinito, ben distante dal massimo numero di archi del grafo, $\frac{\binom{n}{k}^2}{2}$, rendendo quindi conveniente una rappresentazione come liste di adiacenza. Il costo di determinare se due nodi sono adiacenti in questo caso è $\mathcal{O}((n-k)k)$, in quanto ogni nodo ne ha esattamente $k(n-k)$ adiacenti. Considerando il fatto che è possibile ricavare efficientemente la relazione di adiacenza tra due nodi, o trovare tutti gli adiacenti ad un nodo dato, è anche possibile evitare del tutto di stoccare le adiacenze. Determinare se due nodi sono adiacenti richiede in questo caso al più $\mathcal{O}(k \log(k))$ operazioni: se i k elementi che etichettano un nodo sono rappresentati in ordine, il tempo per determinare se le etichette differiscono per un solo elemento è $\mathcal{O}(k)$, altrimenti li si può prima ordinare in $\mathcal{O}(k \log(k))$.