Algoritmi e Principi dell'Informatica

Tema d'esame del 25 Agosto 2021

Esercizio 1 (8 punti, si svolgano solo i punti a e c in caso di riduzione del 30%)
Si definisca *riconoscitore ad automi gemelli* (RAG) un meccanismo di calcolo ottenuto facendo operare due automi sullo stesso nastro di ingresso. Gli automi operano indipendentemente sul nastro di ingresso, ognuno di essi accede al nastro attraverso una propria testina. La parola presente sul nastro di ingresso si considera riconosciuta se e solo se i due automi la riconoscono (non necessariamente nello stesso numero di mosse). Si consideri il caso dei RAG costruiti come segue e si dica se il loro potere riconoscitore è maggiore, minore o uguale a quello delle loro componenti:

a) RAG costituito da un automa a stati finiti deterministico e uno non deterministico;

b) RAG costituito da un automa a stati finiti deterministico ed un automa a pila deterministico;

c) RAG costituito da due automi a pila non deterministici.

Soluzione

Detti $\mathcal{A}_1$ e $\mathcal{A}_2$ gli automi che compongono un dato riconoscitore ad automi gemelli $\mathcal{A}$, si osserva che il riconoscitore ad automi gemelli riconosce il linguaggio $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. È infatti sufficiente osservare che i due automi che lo compongono operano senza influenzarsi e la condizione di accettazione richiede che entrambi accettino la parola sul nastro d'ingresso. Si ha quindi che

a) Il riconoscitore ad automi gemelli ha la stessa potenza riconoscitiva delle sue componenti: i linguaggi regolari sono chiusi per intersezione

b) Il riconoscitore ad automi gemelli ha la stessa potenza riconoscitiva di un automa a pila deterministico. È possibile derivare questo fatto sia dal fatto che i linguaggi liberi dal contesto sono chiusi rispetto all'intersezione con i linguaggi regolari. Alternativamente, è possibile costruire un riconoscitore a pila deterministico equivalente al riconoscitore ad automi gemelli modificando l' organo di controllo dell'APD per incorporare anche il comportamento dell'automa a stati finiti

c) Il riconoscitore ad automi gemelli ha potenza riconoscitiva superiore ad un automa a pila deterministico. È infatti noto che i linguaggi liberi dal contesto non sono chiusi per intersezione. Un esempio di quanto detto è il riconoscitore ad automi gemelli con $L(\mathcal{A}_1) = a^n b^n c^*$ e $L(\mathcal{A}_2) = a^* b^n c^n$, con $n \in \mathbb{N}$.

Esercizio 2 (8 punti, si svolga solo il punto 1 in caso di riduzione del 30%)

1. Si consideri la seguente funzione e si dica, motivando adeguatamente la risposta, se essa sia calcolabile:
$$w(x,y) = \begin{cases} f_y(x) & \text{se } f_y(y) = f_x(x) \neq \perp \\ \perp & \text{altrimenti} \end{cases}$$

2. Si dica, motivando adeguatamente la risposta, se il seguente insieme T è ricorsivo:
$$T = \{i \in \mathbb{N} \mid M_i \text{ possiede } i \text{ stati senza transizioni uscenti}\}.$$

Soluzione

1. La funzione è calcolabile: una MT che la calcola non deve fare altro che calcolare $f_y(y)$ e $f_x(x)$: se entrambi terminano, controlla se sono uguali e restituisce il valore opportuno. Se uno dei due non termina, la MT non deve terminare.

2. T è ricorsivo: basta costruire M_i tramite l'enumerazione $\mathcal{E}$, e contare il numero di stati con la caratteristica richiesta.

Esercizio 3 (9 punti, si svolga solo il punto 1 in caso di riduzione del 30%)
Si definisca *permutazione bidimensionale* una funzione biunivoca dall'insieme S di tutte le coppie ordinate di numeri interi tra 0 e $n-1$, per un dato intero n, a se stesso.

Si consideri una matrice M quadrata, di lato n. Ogni elemento della matrice è una coppia di numeri interi, x e y, compresi tra 0 e $n-1$. Si interpreti la matrice come una funzione $f(x,y) : (x,y) \mapsto M[x,y]$ con x e y interi, $0 \leq x < n$, $0 \leq y < n$.

1. Si realizzi in pseudocodice un algoritmo a complessità temporale minima che verifica se una matrice M rappresenta una permutazione bidimensionale, dimostrandone la minimalità della complessità.

2. Viene detto *orbita* in una permutazione un (sotto-)insieme degli elementi permutati tale per cui, applicando iterativamente la permutazione ognuno di essi va ad occupare, via via ad ogni iterazione, il posto di tutti gli altri dell'orbita. Si descriva un algoritmo che determina la dimensione dell'orbita massima della funzione rappresentata da una matrice M, assumendo che essa sia una permutazione bidimensionale.

Soluzione

```
1  bool verifica_perm(int M[n][n]){
2     bool visto[n][n] = {{false}};
3     for(int i=0; i<n; i++){
4        for(int j=0; j<n; j++){
5            if (visto[M[i][j].x][M[i][j].y] == true)
6               return false
7            else
8               visto[M[i][j].x][M[i][j].y] = true
9        }
10    }
11    return true
12 }
```

La soluzione ha complessità $\mathcal{O}(n^2)$, minima in quanto è necessario almeno leggere l'intera matrice in input allo scopo di verificare la biiettività della funzione rappresentata.

```
1  bool orbita_max(int M[n][n]){
2     int orbita,orbitamax = 0;
3     bool visto[n][n] = {{false}};
4     for(int i=0; i<n; i++){
5        for(int j=0; j<n; j++){
```

```
6           if (visto[M[i][j].x][M[i][j].y] == false){
7            int curr_i = M[i][j].x;
8            int curr_j = M[i][j].y;
9            int orbita = 1;
10           while( !(curr_i == i && curr_j == j) ){
11              visto[M[curr_i][curr_j].x][M[curr_i][curr_j].y] = true;
12              int tmp = curr_i;
13              curr_i = M[curr_i][curr_j].x;
14              curr_j = M[tmp][curr_j].y;
15              orbita++;
16           }
17           if (orbita > orbita_max) {
18              orbita_max = orbita;
19           }
20          }
21       }
22    }
23    return true
24 }
```

La soluzione ha complessità $\mathcal{O}(n^2)$, sfruttando il fatto che ogni elemento può appartenere ad una sola orbita a causa della biunivocità della permutazione. Come sopra, la complessità è minima in quanto è necessario almeno leggere l'intera matrice in input allo scopo di verificare a quale orbita appartenga ciascuno degli elementi. E' possibile ottenere un guadagno (costante) arrestando la ricerca dell'orbita massima quando gli elementi ancora da ispezionare sono in numero minore di quelli dell'orbita massima trovata fino a quel punto.

Esercizio 4 (7 punti)

Si descriva in maniera esauriente una MT a nastro singolo che accetti il linguaggio $\{a^i b^{i+j} a^j \mid i > 5, j > 3\}$ e se ne valutino la complessità temporale e spaziale.

Soluzione

Una macchina per il linguaggio richiesto può procedere in questo modo: fa varie passate da sinistra verso destra, cancellando, appena le trova, una a poi una b. Usando gli stati dell'organo di controllo, la macchina può assicurarsi che il numero delle passate sia maggiore di 5. Finite le a a sinistra, la macchina procede con passate analoghe da sinistra verso destra, cancellando però ad ogni passata una b ed una a. Come prima, usando gli stati dell'organo di controllo, la macchina può assicurarsi che il numero di queste passate sia maggiore di 3.

Sia n la lunghezza della stringa in ingresso. Complessità spaziale: $\mathcal{O}(n)$ (non viene usata altra memoria), complessità temporale: $\mathcal{O}(n^2)$ (ad ogni passata, lunga al più n, si eliminano 2 simboli).