Complexidade Computacional

Primeira lista de exercícios

Exercício 1 (4.1.1 LP) [1]
Considere a máquina de Turing M dada pela tabela:

q	σ	$\delta(q,\sigma)$
s	a	$(q_1, b, -)$
s	b	$(q_1, a, -)$
s	$\sqcup$	$(h, \sqcup, -)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$

q	σ	$\delta(q,\sigma)$
q_1	a	$(s, a, \rightarrow)$
q_1	b	$(s, b, \rightarrow)$
q_1	$\sqcup$	$(s, \sqcup, \rightarrow)$
q_1	$\triangleright$	$(q_1, \triangleright, \rightarrow)$

(a) Descreva a computação de M começando da configuração $(s, \triangleright, aabbba)$.

(b) Descreva informalmente o que M faz.

(c) Descreva, dando uma tabela como a acima, uma máquina de Turing mais simples que faz a mesma tarefa.

Exercício 2 (4.2.2 LP)
Descreva máquinas de Turing que *decidam* as seguintes linguagens sobre o alfabeto $\{a, b\}$:
(i) $\emptyset$; (ii) $\{\epsilon\}$; (iii) $\{a\}$; (iv) $\{a\}^*$.

Exercício 3 (4.2.3 LP)
Descreva uma máquina de Turing que *aceite* a linguagem a^*ba^*b.

Exercício 4 (2.8.8) (este foi a lista 0)
Suponha que máquinas de Turing podem *remover* e *inserir* símbolos nas fitas, ao invés de somente escrever.

(a) Defina cuidadosamente a função de transição e a computação de tal máquina.

(b) Mostre que se tal máquina opera em tempo $f(n) \geq n + 2$, então ela pode ser simulada por uma máquina de Turing comum operando em tempo $O(f^2(n))$.

Exercício 5
Descreva e calcule a complexidade em espaço e tempo de uma máquina de Turing para calcular:
(a) $\lceil \log_2 n \rceil$; (b) $n + m$.
No item (a) considere que a entrada é a codificação binária de um inteiro $n > 0$, e que essa representação não contenha 0's à esquerda.
No item (b), considere que a entrada é codificada da forma

$$\triangleright x + y$$

onde x e y são as representações binárias de n e m e o $+$ é um símbolo especial.

Exercício 6 (2.8.11)
(**Linguagens regulares**) Suponha que uma linguagem L é decidida por uma máquina de Turing com entrada e saída que usa espaço constante.

[1] LP = livro Elementos de Teoria da Computação do Lewis e Papadimitriou

(a) Mostre que existe uma máquina de Turing equivalente que decide L com apenas um string de entrada "read-only".

Tal máquina é chamada de um *two-way finite automaton*. Um *autômato finito* é um two-way finite automaton como movimento de cursor somente para a direita ($\rightarrow$). Uma linguagem decidida por um autômato finito é dita *regular*.

(b) Mostre que a seguinte linguagem é regular:

$$L = \{x \in \{0,1\}^* | x \text{ contém pelo menos dois 0s mas não dois 0s consecutivos}\}.$$

Exercício 7
Considere o seguinte algoritmo para multiplicação de matrizes cujas entradas são inteiros não-negativos. A entrada do algoritmo são duas matrizes

$$A = [a_{ij}]_{n\times n} \text{ e } B = [b_{ij}]_{n\times n}.$$

A saída é uma matriz $C = [c_{ij}]_{n\times n}$. O Algoritmo é o seguinte:

Passo 1 Calcule

$$a = \max_{\substack{1\le i\le n\\ 1\le j\le n}} \{a_{ij}\}, \qquad b = \max_{\substack{1\le i\le n\\ 1\le j\le n}} \{b_{ij}\} \quad \text{e} \quad x = nab + 1.$$

Passo 2 Calcule

$$b_k = \sum_{j=1}^{n} b_{kj} x^{j-1} \text{ para } k = 1, \ldots, n.$$

Passo 3 Calcule

$$c_i = \sum_{k=1}^{n} a_{ik} b_k \text{ para } i = 1, \ldots, n.$$

Passo 4 Calcule

$$c_{i1} = c_i \bmod x \quad \text{e} \quad q_{i1} = c_i \div x, \text{ para } 1 \le i \le n,$$

onde mod e $\div$ são, respectivamente, os operadores resto da divisão e quociente.

Passo 5 Calcule

$$c_{ij} = q_{i(j-1)} \bmod x \quad \text{e} \quad q_{ij} = q_{i(j-1)} \div x, \text{ para } 1 \le i \le n \text{ e } 2 \le j \le n.$$

Calcule o número de somas, multiplicações e divisões que o algoritmo faz para multiplicar duas matrizes $n \times n$. Qual é a complexidade desse algoritmo, em função de n?

No livro de análise de algoritmos do Cormen, Leiserson e Rivest é mencionado o seguinte.

> By using advanced techniques beyond the scope of this text, one can in fact multiply $n \times n$ matrices in better than $\Theta(n^{\lg 7})$ time. The current best upper bound is approximately $O(n^{2.376})$. The best lower bound known is just the obvious $\Omega(n^2)$ bound (obvious because we have to fill in n^2 elements of the product matrix). Thus, we currently do not know how hard matrix multiplication really is.

Compare e comente os resultados que você obteve e com a observação acima.
Obs.: O fato do algoritmo acima só funcionar para matrizes de inteiros não-negativos não é problema. É fácil decompor qualquer matriz inteira como a subtração de duas matrizes, cada uma sendo formada somente de inteiros não-negativos.