Lista 4 — Algoritmos Probabilísicos **[JD 24-3]**

Exercício 1. Escreva um programa RAM para o prolema de paridade.

Exercício 2. Prove que

1. Se $\mathsf{NP} \neq \mathsf{coNP}$ então $\mathsf{P} \neq \mathsf{NP}$.
2. Se $\mathsf{NP} \subseteq \mathsf{BPP}$ então $\mathsf{NP} = \mathsf{RP}$.

Exercício 3. Sejam M uma máquina de Turing probabilística e L uma linguagem tais que para constantes $0 < \varepsilon_1 < \varepsilon_2 < 1$ e todo $w \in \{0,1\}^*$

1. se $w \in \mathsf{L}$, então $\mathbb{P}[\,\mathsf{M}(w) = 1\,] \geqslant \varepsilon_2$, e
2. se $w \notin \mathsf{L}$, então $\mathbb{P}[\,\mathsf{M}(w) = 1\,] < \varepsilon_1$;

prove que L está em BPP.

Exercício 4. A classe PP — *Probabilistic Polynomial time* — é a classe das linguagens L para as quais existe um algoritmo probabilístico M de tempo polinomial tal que

$$\mathbb{P}[\,\mathsf{M}(w) \neq \mathsf{L}(w)\,] < 1/2, \text{ para todo } w \in \{0,1\}^*.$$

Prove as seguintes inclusões

1. $\mathsf{NP} \subseteq \mathsf{PP}$.
2. $\mathsf{BPP} \subseteq \mathsf{PP}$.

Exercício 5. Suponha que exista um algoritmo determinístico eficiente A que, dado inteiros a e $\mathsf{N} = pq > 1$, para p e q primos distintos, determina de modo eficiente uma solução de $x^2 \equiv a \pmod{\mathsf{N}}$. Sorteie $b \in \mathbb{Z}_{\mathsf{N}}^*$ e compute $b^2 \bmod \mathsf{N}$, em seguida use o algoritmo A para determinar uma solução x de $\mathsf{X}^2 \equiv b^2 \pmod{\mathsf{N}}$. Verifique que com probabilidade 1/2 temos $x \not\equiv \pm b \pmod{\mathsf{N}}$. Prove que nesse caso $\mathrm{mdc}(b - x, \mathsf{N}) \in \{p, q\}$. Conclua que se é possível determinar raiz quadrada módulo N de maneira eficiente então é possível fatorar N de maneira eficiente.

Exercício 6. Considere um sistema de codificação (como descrito no cap. 1 das notas de aula) com funções de codificação $\mathsf{E}_k(w)$ e de decodificação $\mathsf{D}_k(w)$, ambas computáveis em tempo polinomial e usam chave k menor que o texto w. A chave k é uma chave aleatória (sorteada uniformemente de um conjunto K de chaves) compartilhada entre as partes e $\mathsf{D}_k(\mathsf{E}_k(w)) = w$ para todo texto w.

Prove que se $\mathsf{P} = \mathsf{NP}$ então existe um algoritmo A de tempo polinomial tal que para todo w, existem $x_0, x_1 \in \{0,1\}^{|w|}$ com

$$\mathop{\mathbb{P}}_{\substack{b \in_{\mathrm{R}} \{0,1\} \\ k \in_{\mathrm{R}} \{0,1\}^n}} [\mathsf{A}(\mathsf{E}_k(x_b)) = b] \geqslant \frac{3}{4}$$

em que $n < |w|$. (*Dica*: tome S o conjunto das sequências binárias y tais que $y = \mathsf{E}_k(0^m)$ para algum k; $\mathsf{A}(y) = 0$ se $y \in \mathsf{S}$ e $\mathsf{A}(y) = 1$ caso contrário.)

Exercício 7. Uma rede é dada por um grafo nãoo-dirigido G onde os vértices sã0 os processadores e as arestas os fios. São dados N pacotes e, para cada pacote, são dados a origem o destino e a rota. Em cada aresta trafega um pacote por passo. Um *escalonamento* para um conjunto de pacotes especifica o *tempo* dos movimentos do pacote pelas suas rotas, ou seja, especifica quais pacotes se movem e quais esperam em cada passo. O objetivo é produzir um escalonamento que minimiza o tempo total e o tamanho máximo de um buffer necessários para o roteamento.

(a) A dilatação d é a maior distância percorrida por um pacote, o congestionamento c é o número máximo de pacotes que passa por uma aresta durante todo o roteamento. Argumente que o tempo requerido por qualquer escalonamento é pelo menos $\Omega(c + d)$.

(b) Considere o seguinte escalonamento onde muitos pacotes podem usar uma determinada aresta durante um único passo. Atribua a cada pacote um *atraso*: um número inteiro escolhido aleatoriamente, de forma uniforme e independente, no intervalo $[1, \lceil \alpha / \log(Nd) \rceil]$, onde α é uma constante. Um pacote que tenha recebido um atraso x espera no seu vértice de origem por x passos; depois, ele se move pela sua rota até o seu destino sem parar. Dê um limitante superior para a probabilidade de mais de $O(\log(Nd))$ pacotes usarem uma aresta e no instante t.

(c) Nas condições do item anterior, mostre que a probabilidade de mais que $O(\log(Nd))$ pacotes passarem por qualquer aresta em qualquer instante é, no máximo, $1/(nd)$ se α é suficientemente grande.

(d) Use os itens anteriores para projetar um algoritmo (simples) que produza um escalonamento de tamanho $O(c + d \log(Nd))$, utilizando buffers de tamanho $O(\log(Nd))$ e seguindo a restrição de no máximo um pacote por aresta em cada passo.