Atividade de Projeto e Análise de Algoritmos
Estudante: Guilherme Brizzi

1. **Prove formalmente que (usando a definição de Θ) que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$**

A definição formal da notação assintótica Θ enuncia que, dada uma função f(n), ela pertence ao grupo Θ(g(n)) se existem constantes $c_1, c_2 \in \mathbb{R}^+$ tal que

$$c_1 \times g(n) \leq f(n) \leq c_2 \times g(n),$$

para n de tamanho considerável.

Dessa forma, para provar que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$, devemos encontrar c_1 e c_2 que satisfaçam a inequação.

Nesse caso, tomemos $c_1 = \frac{1}{14}$, $c_2 = 1$:
Para $\forall n \geq 7$,

$$c_1 \times g(n) \leq f(n)$$

Pois, para n = 7

$$\frac{1}{14} \times n^2 \leq \frac{1}{2}n^2 - 3n$$
$$\frac{1}{14} \times 7^2 \leq \frac{1}{2}7^2 - 3 \times 7$$
$$\frac{1}{14} \times 49 \leq \frac{1}{2}49 - 3 \times 7$$
$$\frac{7}{2} \leq \frac{49}{2} - \frac{42}{2}$$
$$\frac{7}{2} \leq \frac{7}{2}$$

Analogamente para c_2, para $\forall n \geq 1$,

$$f(n) \leq c_2 \times g(n)$$

Pois, para n = 1

$$\frac{1}{2} \times n^2 \leq 1 \times n^2$$
$$\frac{1}{2} \times 1^2 \leq 1^2$$
$$\frac{1}{2} \leq 1$$

Assim, fica provado que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$.

2. $\boldsymbol{n^2 = O(n^3)}$?

Suponha-se que $n^2 = O(n^3)$. Para isso ocorrer, deve haver uma constante c, tal que $f(n) \leq c \times g(n)$, ou seja, $n^2 \leq c \times n^3$.

Tome-se $n \geq 1, c = 1$:

$$n^2 \leq c \times n^3$$
$$1^2 \leq 1 \times 1^3$$
$$1 \leq 1$$

Assim, fica provado que $n^2 = O(n^3)$.

3. **Explique por que a declaração "O tempo de execução no algoritmo A é no mínimo $O(n^2)$" é isenta de significado.**

Essa declaração é isenta de significado, pois a notação O() é relativa ao limite superior da notação assintótica. Portanto, não faz sentido se falar em tempo mínimo em termos de O().

4. **a) É verdade que $\boldsymbol{2^{n+1} = O(2^n)}$?**

Sim, é verdade. Isso ocorre pois é possível encontrar uma constante c, tal que $f(n) \leq c \times g(n)$, ou seja, $2^{n+1} \leq c \times 2^n$. Tome $c = 2, n \geq 1$:

$$2^{n+1} \leq 2 \times 2^n$$
$$2^{n+1} \leq 2^{n+1}$$

Está provado que $2^{n+1} = O(2^n)$.

b) É verdade que $\boldsymbol{2^{2n} = O(2^n)}$?

Não, é falso. Isso ocorre pois não é possível encontrar uma constante c, tal que $f(n) \leq c \times g(n)$, ou seja, $2^{2n} \leq c \times 2^n$. Para satisfazer essa inequação, c teria de tomar o valor de 2^n. Porém, n é uma variável. Então, não é possível encontrar uma constante c.

Está provado que $2^{2n} \neq O(2^n)$.