# Lista de Exercícios 02 – Unidade 04

Nome: Lucca Magnino – 202401805

## Questões discursivas:

1) O que significa dizer que uma função g(n) é O(f(n))?

   Significa que f(n) é o limite superior de g(n), ou seja, cresce até f(n).

2) O que significa dizer que uma função g(n) é $\Theta$(f(n)) (theta de n)?

   Significa que f(n) é o caso médio para g(n), ou seja, cresce proporcionalmente a f(n).

3) O que significa dizer que uma função g(n) é Ω(f(n)) (ômega de n)?

   Significa que f(n) é o limite inferior para g(n), ou seja, cresce sempre acima ou igualmente a f(n).

4) Dois algoritmos A e B possuem complexidades $n^5$ e 2 n respectivamente. Você utilizaria o algoritmo B ao invés do A, em qual caso? Explique.

   Analisando a complexidade, observamos como o algoritmo B cresce muito mais rápido, em comparação ao A, portanto quando a entrada for pequena, a precisão do B ainda pode ser efetiva, entretanto, em casos que a entrada é maior, o algoritmo A é mais efetivo, visto que não irá "ignorar" uma série de valores.

5) Pesquise sobre quais problemas costumam ser exponenciais. Comente sobre dois deles.

   **Problema do Caixeiro Viajante:** Esse problema é baseado na busca pelo caminho mais curto que passa por todas as cidades de um conjunto exatamente uma vez e retorna à cidade de origem.
   **Satisfação Boolena:** O problema é baseado em determinar se existe uma atribuição de valores verdadeiro ou falsos que satisfaça a fórmula booleana.

6) Indique a ordem de complexidade, no pior caso, das seguintes funções de custo:

a. $f(n) = 2n + 10 = O(n)$

b. $f(n) = 1/2n * (n + 1) = O(n^2)$

c. $f(n) = 1/2\ n^2 = O(n^2)$

d. $f(n) = 1/2\ n^4 - 3n^2 + 5n + 7 = O(n^4)$

e. $f(n) = 7n + 3\ \log_2(n) + 20 = O(n)$

f. $f(n) = n! + 5n^2 + 10 = O(n!)$

g. $f(n) = 3 * 5000^n + 1000 = O(5000^n)$

h. $f(n) = 10^{10} = O(1)$

7) Calcule a complexidade, no pior caso, do seguinte fragmento de código:

```
int i,j,k;
for(i = 0; i < N; i++){
    for(j=0; j < N; j++){
        R[i][j] = 0;
        for(k=0; k < N; k++)
            R[i][j] += A[i][k] * B[k][j];
    }
}
```

No pior dos casos, teremos os três loops sendo realizados até o fim, ou seja, até N, logo, a complexidade será obtida pelo produto dos laços, ou seja, $O(N^3)$.