**Análise de Algoritmo**
**Lista de Exercício 1**

1. Prove que $\lg(n!) = \Theta(n \lg n)$ e que $n! = o(n^n)$.

2. 

   (a) É $2^{n+1} = O(2^n)$?

   (b) É $2^{2n} = O(2^n)$?

3. Ordene as seguintes funções pela sua ordem de crescimento; isto é, encontre um arranjo $g_1, g_2, \ldots$ das funções satisfazendo $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, etc. Particione sua lista em classes de equivalências tal que $f(n)$ e $g(n)$ estão na mesma classe se e somente se $f(n) = \Theta(g(n))$.

| | | | | | |
|---|---|---|---|---|---|
| $(\sqrt{2})^{\lg n}$, | $n^2$, | $n!$, | $(\lg n)!$, | $(\frac{3}{2})^n$, | $n^3$, |
| $\lg^2 n$, | $\lg(n!)$, | $2^{2^n}$, | $n^{1/\lg n}$, | $\ln \ln n$, | $n \cdot 2^n$, |
| $n^{\lg \lg n}$, | $\ln n$, | $1$, | $2^{\lg n}$, | $(\lg n)^{\lg n}$, | $e^n$, |
| $4^{\lg n}$, | $(n+1)!$, | $\sqrt{\lg n}$, | $2^{\sqrt{2 \lg n}}$, | $n$, | $2^n$, |
| $n \lg n$ | $2^{2^{n+1}}$ | | | | |

4. Uma das questões de uma prova elaborada pelo Prof. Justus consistia em escrever um algoritmo *encontsoma*$(S_1, S_2, x)$, que dado dois vetores $S_1$ e $S_2$ (cada um de tamanho $n$) determinava um par de elementos (se algum), um de $S_1$ e outro de $S_2$, cuja soma fosse igual a um valor $x$, e cujo tempo de execução, no pior caso, tivesse um comportamento da ordem $O(n \lg n)$.

   Um dos alunos do Prof. Justus propôs a seguinte solução:

```
encontsoma(S1,S2,x)
    for i = 1 to n
        for j = 1 to n
            if S1[i]+S2[j]=x
                then
                    return(S1[i],S2[j])
```

   (a) Qual foi a correção do Prof. Justus? Explique o porque?

   (b) Proponha um algoritmo alternativo.

5. Prove a seguinte afirmação: Se $f_1(n) = O(g_1(n))$ e $f_2(n) = O(g_2(n))$ então $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$.

6. Prove a seguinte afirmação: Se $f_1(n) = O(g_1(n))$ e $f_2(n) = O(g_2(n))$ então $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$.

7. Bubblesort é um algoritmo de ordenação frequentemente ensinado nos cursos introdutórios de computação, talvez por causa do seu nome. Aqui está o pseudo-código para o algoritmo:

   ```
   Bubble-sort(A)
       for i ← 2 to length[A] - 1 do
           for j ←1 to length[A] - i do
               if A[j] >A[j+1] then
                   Troque A[j] ⟷ A[j+1]
   ```

   Qual é o tempo de execução deste algoritmo no pior caso e no caso médio?