Lista de Exercícios

1. Simplifique, ou reescreva de forma simplificada, as equações a seguir (quando a base do logarítmo não for indicada, considere base binária (2)): $(\sqrt{2})^{\log n}$, $n^{1/\log n}$, $n^{\log\log n}$, $4^{\log n}$, $2^{\sqrt{2\log n}}$ e 2^{2n+1}.

$$\begin{aligned}
\sqrt{2}^{\log_2 n} &= n^{\log_2\sqrt{2}} = n^{\frac{1}{2}} = \sqrt{n} \\
n^{\frac{1}{\log_2 n}} &== n^{\frac{\log_2 2}{\log_2 n}} = n^{\log_n 2} = 2^{\log_n n} = 2 \\
n^{\log(\log n)} &= (\log n)^{(\log n)} \\
4^{\log n} &= n^{\log 4} = n^2 \\
2^{\sqrt{2\log n}} &= (2^{\sqrt{2}})^{\sqrt{\log n}} \\
2^{2n+1} &= 2\cdot 4^n
\end{aligned}$$

2. Encontre as aproximações de Stirling para $n!$ e para $\log(n!)$.

$$\log(n!) \approx n\log n - \frac{n}{\log_e 2} + \frac{1}{2}\log n + O(1) \qquad \text{livro da Laira, } 2^a \text{ edição, pág.20}$$

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n (1+\Theta(1/n)) \qquad \text{Cormen tradução } 2^a \text{ edição americana, pág 44}$$

3. Prove por indução as seguintes igualdades:

$$\sum_{i=1}^{n} 2^i = 2^{n+1} - 2 \tag{1}$$

$$\sum_{i=0}^{n} i^2 = \frac{n\cdot(n+1)\cdot(2n+1)}{6} \tag{2}$$

$$\sum_{i=0}^{n} i\cdot 2^i = 2 + (n-1)\cdot 2^{n+1} \tag{3}$$

$$\texttt{Hipótese} \sum_{i=0}^{n} i\cdot 2^i = 2 + (n-1)\cdot 2^{n+1}$$

$\texttt{Base:}$ $\texttt{Para n=1, tem-se}$ $1\cdot 2^1 = 2 + (1-1)\cdot 2^{1+1}$

$2 = 2$ Base ok

$\texttt{Objetivo:}2 + (n-1+1)\cdot 2^{(n+1)+1} = 2 + n\cdot 2^{n+2}$

$$\begin{aligned}
\texttt{Passo da indução:} \sum_{i=0}^{n+1} i\cdot 2^i &= \sum_{i=0}^{n} i\cdot 2^i + 2 + (n+1)\cdot 2^{n+1} \\
&= 2 + (n-1)\cdot 2^{n+1} + (n+1)\cdot 2^{n+1} \qquad \text{Usando a hipótese} \\
&= 2 + 2n\cdot 2^{n+1} = 2 + n\cdot 2^{n+2}\blacksquare
\end{aligned}$$

$$\texttt{Hipótese} \sum_{i=1}^{n} 2^i = 2^{n+1} - 2$$

$$\texttt{Base: Para n=1, tem-se } 2^1 = 2^2 - 2$$
$$2 = 2 \qquad \text{Base ok}$$

$$\texttt{Objetivo:} 2^{(n+1)+1} - 2 = 2^{n+2} - 2$$

$$\texttt{Passo da indução:} \sum_{i=0}^{n+1} i^2 = \sum_{i=0}^{n} i^2 + 2^{n+1}$$
$$= 2^{n+1} - 2 + 2^{n+1} \qquad \text{Usando a hipótese}$$
$$= 2 \cdot 2^n - 2 + 2 \cdot 2^n = 2^{n+2} - 2 \blacksquare$$

$$\texttt{Hipótese} \sum_{i=0}^{n} i^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$$

$$\texttt{Base: Para n=1, tem-se } 1^2 = \frac{1 \cdot (1+1) \cdot (2+1)}{6}$$
$$1 = 1 \qquad \text{Base ok}$$

$$\texttt{Objetivo:} \frac{(n+1) \cdot ((n+1)+1) \cdot (2(n+1)+1)}{6} = \frac{(n+1) \cdot (n+2) \cdot (2n+3)}{6}$$

$$\texttt{Passo da indução:} \sum_{i=0}^{n+1} i^2 = \sum_{i=0}^{n} i^2 + (n+1)^2$$
$$= \frac{n \cdot (n+1) \cdot (2n+1)}{6} + (n+1)^2 \qquad \text{Usando a hipótese}$$
$$= \frac{(n+1)}{6} \cdot (2n^2 + 7n + 6) = \frac{(n+1)}{6} \cdot (n+2) \cdot (2n+3) \blacksquare$$

4. Exercício Extra: Considere que você tenha um conjunto N com n números inteiros, não repetidos, enumerados de 1 a n. Seja X e Y dois subconjuntos de N (tendo os valores não ordenados) de tamanhos x e y respectivamente. Projete um algoritmo que determine quantos números têm na intersecção entre X e Y. Qual a complexidade do seu algoritmo? Descreva a idéia do algoritmo.

   Resposta. Os seguintes algoritmos solucionariam o problema. Sem perder generalidade, considero que $x < y$ (caso não fosse, apenas poderia-se chamar o X de Y e vice-versa):

   - para cada item do conjunto X, verifica se o item está no outro conjunto. Complexidade $\Theta(x \cdot y)$. Como $1 \leq x, y \leq n$, a complexidade seria $O(n^2)$.
   - Ordena os dois conjuntos com merge-sort e executa um `scan` dos dois conjuntos ordenados. Complexidade $\mathrm{O}(x \log x)$. Como $1 \leq x \leq n$, a complexidade seria $\mathrm{O}(n \log n)$.
   - Ordena os dois conjuntos com counting-sort e executa um `scan` dos dois conjuntos ordenados. Complexidade $\Theta(n)$.
   - Aloque um vetor b com n boleanos inicializados com zero, e para cada item X_i marque $b[X_i] = \mathit{true}$. Para cada item Y_i, verifique se $b[Y_i] = \mathit{true}$ (neste caso, o item i pertence à intersecção dos dois conjuntos).

   Caso os elementos de X e Y já estiverem ordenados, com um *scan* já pode-se verificar a interseção, e o algoritmo teria complexidade $O(x)$. Como $1 \leq x \leq n$, a complexidade seria $\mathrm{O}(n)$.