NOTAÇÃO ASSINTÓTICA - Aula 2 (Cormen, cap. 3)

1. EXERCÍCIO: prove ou apresente contraexemplo: $n = \Omega(n^2)$.

RESPOSTA:

Nós provaremos que $n \neq \Omega(n^2)$.

Suponha, por absurdo, que $n = \Omega(n^2)$. Logo, existem "c" e "n0" tais que, para todo $n \geq n0$, $c \cdot n^2 \leq n$. São dois casos:

* CASO 1: $c \geq 1$. Nesse caso, considere $n = n0 + 1$. Temos então:

$$\begin{aligned} c \cdot n^2 &\geq n^2 && \text{--> pois } c \geq 1 \\ &= (n0 + 1) \cdot n && \text{--> pois } n = n0 + 1 \\ &= n0 \cdot n + n \\ &> n. && \text{--> pois } n > n0 \geq 1 \end{aligned}$$

* CASO 2: $c < 1$. Nesse caso, seja $n = \text{máx}\{ n0, \text{teto}(1/c) + 1 \}$. Temos então:

$$\begin{aligned} c \cdot n^2 &= c \cdot n \cdot n \\ &\geq c \cdot (\text{teto}(1/c) + 1) \cdot n && \text{--> pela definição de "n"} \\ &> c \cdot \text{teto}(1/c) \cdot n \\ &\geq c \cdot 1/c \cdot n \\ &= n. \end{aligned}$$

Pela análise acima, portanto, existe $n \geq n0$ tal que $c \cdot n^2 > n$, um absurdo.
Logo, a hipótese inicial de que $n = \Omega(n^2)$ é falsa, CQD.

---

2. DEFINIÇÃO: Dadas funções f e g de N (Naturais) em R (reais), nós dizemos que $f = \Theta(g)$ sse existem reais positivos c_1 e c_2 e um natural positivo n_0 tais que $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ para todo $n \geq n_0$.

3. EXERCÍCIO: mostre que $n = \Theta(5n)$.

4. EXERCÍCIO: mostre que $f = \Theta(g)$ sse $f = \Omega(g)$ e $f = O(g)$.

5. EXERCÍCIO: prove ou refute: para quaisquer constantes reais "a" e "b" e funções "f" e "g", $f = \Theta(a \cdot g + b)$ sse $f = \Theta(g)$.

6. DEFINIÇÃO: Dadas funções f e g de N (Naturais) em R (reais), nós dizemos que $f = o(g)$ sse para QUALQUER constante real positiva c, existe n_0 tal que $0 \leq f(n) < c \cdot g(n)$ para todo $n \geq n_0$.

7. DEFINIÇÃO: Dadas funções f e g de N (Naturais) em R (reais), nós dizemos que $f = \omega(g)$ sse para QUALQUER constante real positiva c, existe n_0 tal que $0 \leq c \cdot g(n) < f(n)$ para todo $n \geq n_0$.

8. EXERCÍCIO: mostre que $n = \omega(1)$.

==

RESPOSTA:

Sejam $f(n) = n$ e $g(n) = 1$, $\forall n$ natural.
Seja $c > 0$ (real).
Seja $n0$ = teto(c) + 1.
Seja $n >= n0$. Temos:

$$
\begin{aligned}
c*g(n) &= c*1 \\
&= c \\
&<= \text{teto}(c) \\
&< \text{teto}(c) + 1 \\
&= n0 \\
&<= n \\
&= f(n).
\end{aligned}
$$

Logo, por definição, $n = \omega(1)$, CQD.

OBSERVAÇÃO: também é necessário mostrar que $c*g(n) >= 0$, mas isso é imediato,
---------- pois $c > 0$ (por essa razão, nós tipicamente omitimos essa parte da demonstração).

==

9. EXERCÍCIO: mostre que se $f = \theta(g)$, então $f \mathrel{!=} o(g)$.

10. EXERCÍCIO: mostre que $f = o(g)$ sse $g = \omega(f)$.

11. OBSERVAÇÕES: veja abaixo detalhes envolvendo as notações assintóticas; o
----------- algoritmo de ordenação por inserção é utilizado como auxílio.

a) A ordenação por inserção executa em tempo $O(n)$ NO MELHOR CASO,
e em tempo $O(n^2)$ NO PIOR CASO.

b) Observe que a ordenação por inserção executa, no melhor caso,
também em tempo $O(n^2)$, mas não em tempo $\Omega(n^2)$, e portanto também não em
tempo $\theta(n^2)$. No melhor caso, ela executa em tempo $\theta(n)$.

c) A ordenação por inserção executa em tempo $O(n^2)$ (OBSERVE que o "caso"
não foi mencionado; nesse caso, a afirmação se refere a todos os casos).
Não é correto, entretanto, dizer que esse algoritmo executa
em tempo $\Omega(n^2)$, pois há casos (como no melhor caso) em que ele executa
em $O(n)$.

d) A busca linear executa em tempo $O(n)$ (logo, também em $O(n^2)$, $O(n^3)$, etc).
A busca linear executa em tempo $\Omega(1)$.
A busca linear executa, no MELHOR CASO, em tempo $O(1)$.
(Logo, a busca linear não executa em tempo $\Omega(n)$.)
Logo, a busca linear executa em tempo $\theta(1)$ NO MELHOR CASO.

e) Considere o algoritmo trivial abaixo, que recebe como entrada um vetor de
"n" números reais e que retorna o maior dentre os dois primeiros
elementos do vetor. Se $t(n)$ é a função que informa o tempo de execução
desse algoritmo, então observe que $t = O(n)$. Obviamente, $t = O(1)$ também
é verdade, e portanto não é verdade que $t = \Omega(n)$, e nem que $t = \Omega(\lg n)$,
etc. Também é verdade que $t = \Omega(1)$, e portanto que $t = \theta(1)$.

```
=========================================================
Algoritmo: maior_dos_2_prim
Entrada: um vetor A[1..n] de números reais, com n >= 2.
Saída: um número real.
---------------------------------------------------------
SE A[1] >= A[2]
| RETORNE A[1]
SENÃO
| RETORNE A[2].
=========================================================
```

TEMPO DE EXECUÇÃO DE ALGORITMOS RECURSIVOS

Considere a versão recursiva do algoritmo de busca binária:

```
=================================================================
Algoritmo: busca_bin_rec
Entrada: um vetor ORDENADO A[1..n] de números reais, e um real x.
Saída: um natural de 1 a n+1.
-----------------------------------------------------------------
RETORNE busca_bin_rec_aux(A, x, 1, n).
=================================================================
```

```
=================================================================
Algoritmo: busca_bin_rec_aux
Entrada: (1) um vetor ORDENADO A[1..n] de números reais,
         (2) um real x,
         (3) um natural i,
         (4) um natural j.
Saída: um natural de 1 a n+1.
-----------------------------------------------------------------
SE j < i
| RETORNE n+1.
m := (i+j) div 2.
SE x = A[m]
| RETORNE m.
SE x < A[m]
| RETORNE busca_bin_rec_aux(A, x, i, m-1).
SENÃO
| RETORNE busca_bin_rec_aux(A, x, m+1, j).
=================================================================
```

Observe que f(t) limita superiormente o tempo de execução de busca_bin_rec_aux para um intervalo de tamanho "t" -- ou, mais precisamente, t(i,j) --, onde

```
t(i,j) = j -i +1, se i <= j
t(i,j) = 0,       em caso contrário

f(t)   = a + f(piso(t/2)), se t > 0
f(0)   = b
```

e "a" e "b" são naturais positivos grandes o suficiente.

12. EXERCÍCIO (ESSENCIAL): mostre que f(t) = O(lg n). Para tanto, você deverá provar primeiramente, por indução, que

```
f(t) <= c + d*teto(lg t) , ou
f(t) <= c + d*(lg t)     , ou
f(t) <= c + d*piso(lg t) ,
```

para todo t >= 0 e para valores adequados de "c" e "d" (em função de "a" e "b").