Estrutura de Dados
Professor: Thiago Queiroz
Aluno: Danilo Alexandrino de Miranda
Matrícula: 20152045050382

→ Lista de Complexidade

01. Limite assintótico superior: $O(f(n))$
Uma função $g(n) = O(f(n))$, se $g(n) \leq c_1 * f(n)$
para uma constante $c_1 > 0$ e para $n \geq n_0$

02. Limite assintótico estrito: $\Theta(f(n))$
Uma função $g(n) = \Theta(f(n))$, se $c_1 * f(n) \leq g(n) \leq c_2 * f(n)$
para constantes c_1 e $c_2 > 0$ e para $n \geq n_0$.

03. Limite assintótico inferior: $\Omega(f(n))$
Uma função $g(n) = \Omega(f(n))$, se $g(n) \geq c_1 * f(n)$
para uma constante $c_1 > 0$ e para $n \geq n_0$

04. $n^2 - n + 549 = 49n + 49 \Rightarrow$
$n^2 - 50n + 500 = 0$
$n_1 = 13,819$
Para $n = 14$

05. $\Theta(n^3)$

06. $2n^3 + 5 = \Theta(n^3)$
Se $f(n) = \Theta(g(n)) = \Theta(n)$, isto é, $g(n) = n$
Seguindo a definição de limite assintótico
$f(n) = 2n^3 + 5$

07. Utilizaria o algoritmo B no lugar do A quando n for maior que 1. Pois quando for $n = 1$, o A é melhor que o B.

08. a) 1. Para $i \leftarrow 1 \ldots n$ faça
2. Para $j \leftarrow 1 \ldots 2^n$ faça
3. } Operações de tempo
4. } constantes

Como j é 2^n para cada iteração de i, as linhas 3 e 4 são executadas um número $\leq n\,2^n$ vezes. Logo o algoritmo é $O(n\,2^n)$

A explicação anterior não está errada, mas $n \cdot 2^n$ é um exagero. Podemos fazer melhor:

$i=1,\ j \leftarrow 1 \ldots 2^1 \to 2^1$ vezes
$i=2,\ j \leftarrow 1 \ldots 2^2 \to 2^2$ vezes
$i=n,\ j \leftarrow 1 \ldots 2^n \to 2^n$ vezes

$2^1 + 2^2 + 2^3 + \ldots + 2^n$ (PG de razão 2)

$= \dfrac{2(2^n - 1)}{2-1} = \underset{\text{constante}}{2} \cdot 2^n - \underset{\text{constante}}{2}$ é $O(2^n)$

b) $S = 1^2 + 2^2 + \ldots + n^2 = \dfrac{n(n+1)(2n+1)}{2}$

↳ soma dos n primeiros quadrados

Agora para o algoritmo:

$i = 1:\ j = 1 \ldots 2^1$
$\quad 1^2 + 2^{1 \cdot 2}$

$i = 2:\ j = 1 \quad 2 \ldots 2^2$
$\quad 1^2 + 2^2 + \ldots + 2^{2 \cdot 2}$

i genérico: $j = 1 \quad 2 \quad 2^i$
$\quad 1^2 + 2^2 + \ldots + 2^{i \cdot 2} \Big\} \dfrac{2^i(2^i+1)(2^{i+1}+1)}{6}$ é $O(2^{3i})$

$i = n:\ j = 1 \quad 2 \ldots 2^n$
$\quad 1^2 + 2^2 + \ldots + 2^{n \cdot 2}$

$\leq \sum_{i=1}^{n} 2^{3i} = \sum_{i=1}^{n} 8^i$ PG de razão 8

$\leq \dfrac{8(8^n - 1)}{8-1}$ é $O(8^n)$

c) 1. para $i = 1 \ldots n$ faça
2. para $j = i \ldots n$ faça
3. operações $\Theta(2^i)$

Para o algoritmo:

$i = 1;\ j = 2^1 + 2^1 \ldots \to n \cdot 2^n \to n^2 \cdot 2^n \to \Theta(n^2 \cdot 2^n)$

↳ n vezes de iteração ↳ $\Theta(2^n)$ //

$1 \cdot 2^1 + 2 \cdot 2^2 + \ldots + n \cdot 2^n$

d) 1. Para $i = 1, \ldots n$ faça
2. $j = 1$
3. enquanto $j < i$ faça
4. operações $\Theta(2^j)$
5. $j = j + 1$

$j = 2^j \Rightarrow j =$

$i = 1, \; j = 1 \to 2^1$
$i = n, \; j = n \to 2^n$

$(n \cdot 2^n) \cdot n \to n^2 2^n \to \Theta(2^n)$

09. Entrada → vetor de tamanho n

```
maior = 0, seg. maior = 0.
for i = 0 ... n do
    if (vetor[i] > maior)
        maior = vetor[i]
end for
for j = 0 ... n do
    if (vetor[i] != maior)
        seg. maior = vetor[i]
end for
```

Complexidade → $\Theta(n^2)$