Semana-02

Created	@March 2, 2023 4:49 PM
Class	Desenho e análise de Algoritmos
Type	FICHA
Materials	https://www.dcc.fc.up.pt/~pribeiro/aulas/daa2223/praticas/aula02.html

Exercício 1) Comparando Funções

Em cada afirmação indicar verdadeiro e falso:

$$f(n) \in O(g(n))$$

$$f(n) \in \Omega(g(n)$$

$$f(n) \in \Theta(g(n)$$

(a)	$f(n) = 2n^3 - 10n^2$	$g(n) = 25n^2 + 37n$
(b)	$f(n) = 56$	$g(n) = \log_2 30$
(c)	$f(n) = \log_3 n$	$g(n) = \log_2 n$
(d)	$f(n) = n^3$	$g(n) = 3^n$
(e)	$f(n) = n!$	$g(n) = 2^n$
(f)	$f(n) = n!$	$g(n) = n^n$
(g)	$f(n) = n \log n + n^2$	$g(n) = n^2$
(h)	$f(n) = \sqrt{(n)}$	$g(n) = \log_2 n$
(i)	$f(n) = \log_3(\log_3 n)$	$g(n) = \log_3 n$

$$O(g(n))\text{ ”} <= \text{”} \;||\; \Omega(g(n)\;\;\text{ ”} >= \text{”} \;||\; \Theta(g(n)\text{ ”} == \text{”}$$

Resposta:

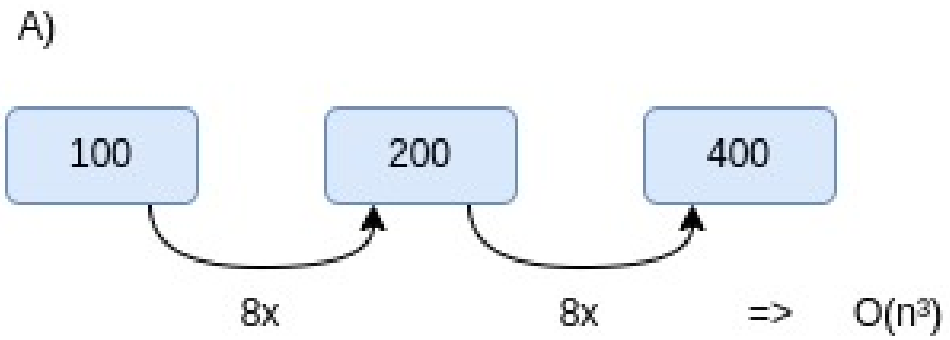
		$f(n) \in O(g(n))$	$f(n) \in \Omega(g(n))$	$f(n) \in \Theta(g(n))$
(a)	$f(n) = 2n^3 - 10n^2 \; ; g(n) = 25n^2 + 37n$	falso	verdadeiro	falso
(b)	$f(n) = 56 \; ; g(n) = \log_2 30$	verdadeiro	verdadeiro	verdadeiro
(c)	$f(n) = \log_3 n \; ; g(n) = \log_2 n$	verdadeiro	verdadeiro	verdadeiro
(d)	$f(n) = n^3 \; ; g(n) = 3^n$	verdadeiro	falso	falso
(e)	$f(n) = n! \; ; g(n) = 2^n$	falso	verdadeiro	falso

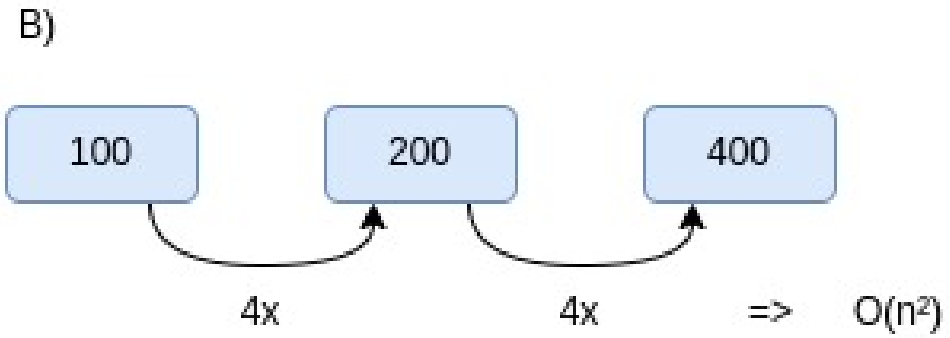
Exercício 2) Taxa de crescimento e previsão de tempo de execução

Algoritmo	*n=100*	*n=200*	*n=300*	*n=400*	*n=500*
A	0.003s	0.024s	0.081s	0.192s	0.375s
B	0.040s	0.160s	0.360s	0.640s	1.000s

a. Qual algoritmo é mais eficiente? Justifique a sua resposta.
b. Indique uma estimativa do tempo que cada algoritmo iria demorar para *N=5000*

Resposta a)





Resposta b)

$t(n2) = f(n2)/f(n1) * t1$

$n2 = 5000$

$A: t(500) = 5000^3/100^3 * 0.003$

Exercício 3) Complexidade de ciclos

Para cada um dos seguintes pedaços de código a seguir indicados, indique qual a complexidade temporal:

a)

```
for (int i=0; i<n; i++)
   for (int j=0; j<n; j++)
      count++;
for (int i=0; i<n; i++)
      count++;
```

$$Resposta: \theta(n^2)$$

b)

```
for (int i=0; i<n/2; i++)
   for (int j=0; j<42; j++)
      for (int k=n; k<n+5; k++)
         count++;
```

$$Resposta: \theta(n)$$

c)

```
for (int i=0; i<n; i+=2)
   for (int j=1; j<n; j*=2)
      count++;
```

$$Resposta: \theta(n\ log\ n)$$

Exercício 4) Complexidade de funções recursivas

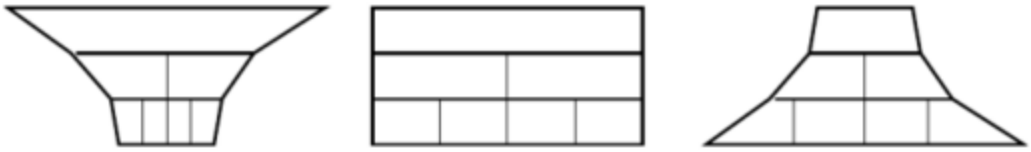
Master Theorem - Uma versão prática

Uma recorrência $\mathbf{aT(n/b) + cn^k}$ ($a \geq 1, b > 1, c$ e k são constantes positivas) resolve para:

(1) $T(n) = \Theta(n^k)$ se $a < b^k$
(2) $T(n) = \Theta(n^k \log n)$ se $a = b^k$
(3) $T(n) = \Theta(n^{\log_b a})$ se $a > b^k$

Se pensarem na árvore de recursão, intuitivamente, estes 3 casos correspondem a:

- (1) O tempo é dominado pelo **primeiro nível**
- (2) O tempo está (uniformemente) **distribuído** por todos os níveis
- (3) O tempo é dominado pelo **último nível**



a)

```
int f1(int v[], int a, int b) {
  if (a>=b) return v[a];
  int m = (a+b) / 2;
  return v[b] + f1(v, a, m);
}
```

$T(n) = T(n/2) + \theta(1)$

$a = 1; b = 2; k = 0$

$a = b^k \ (1 = 2^0)$

$T(n) = \theta(n^k \log n) = \theta(\log n)$

b)

```
int f2(int v[], int a, int b) {
  if (a>=b) return v[a];
  int m = (a+b) / 2;
```

```
  return f2(v, a, m) + f2(v, m+1,b);
}
```

$T(n) = 2T(n/2) + \theta(1)$

$a = 2; b = 2; k = 0$

$a > b^k$

$T(n) = \theta(n \log b^a) = \theta(n \log 2^2) = \theta(n)$

....