\zzabarito_Exercicios_1

1. É o 3., dos Exercícios (transparência 32). Um algoritmo tem complexidade 2^n. Num certo computador **A**, em um tempo *t*, o algoritmo resolve um problema de tamanho 25. Imagine agora que você tem disponível um computador **B** 128 vezes mais rápido. Qual é o tamanho máximo do problema que o mesmo algoritmo resolve no mesmo tempo *t*?

   $C = 2^n$

   $T = C/Vc$, nesse caso $Ta = Tb$ e $Vcb = 128Vca$
   Em A, n = 25 ? em B quanto será n ??
   logo

   $2^{25} /Vca = 2^n/128Vca$

   $2^n = 1282^{25}$

   $2^n = 2^72^{25}$

   $2^n = 2^{7+25}$

   $2^n = 2^{32}$

   n = 32.

   Ou seja, o tamanho em B é 32.

2. Do livro texto clrs, 1.2-1. Cite um exemplo de aplicação que exige conteúdo algorítmico no nível da aplicação e discuta a função dos Algoritmos envolvidos.

   Uma página na internet de busca de hotéis de uma determinada cidade, utiliza algoritmos de busca, comparação e ordenação para apresentar as melhores opções para seus clientes no topo da página.

3. Do livro texto clrs, 1.2-2. Vamos supor que estamos comparando implementações de ordenação por inserção e ordenação por intercalação na mesma máquina. Para entradas de tamanho *n*, a ordenação por inserção é executada em $8n^2$ etapas, enquanto a ordenação por intercalação é executada em $64n \lg n$, onde onde lg *n* representa $\log_2 n$ etapas. Para que valores de *n* a ordenação por inserção supera a ordenação por intercalação?
   Digitei no google tabela com cálculo de logaritmos na base 2

Valores de n	Inserção ($8n^2$)	Intercalação($64n \lg n$)
2	32	128
10	800	3,322x640=2126,08
20	3200	4,3219x1280=5532,02
31	7688	4,9542x1984=9829,1328
41	13448	5,3576x2264 =14058,34
43	14792	5,4263x 2752=14993,17
44	15488	5,4594x2816=15273,67

Logo para valores de n menores ou igual a 43.

Outra solução

$$8n^2 \leq 64nlgn$$

$$n^2 \leq 8nlgn$$

$$n \leq 8lgn$$

$$\frac{n}{lgn} \leq 8$$

$$n \leq 43 \text{ [*]}$$

[*] Resolvido utilizando o algoritmo de força bruta disponível no repositório do github (https://github.com/meitcher/mtcblog/blob/master/CLRS/Cap1/bruteforce.py).

4. Do livro texto clrs, 1.2-3. Qual é o menor valor de *n* tal que um algoritmo cujo tempo de execução é $100n^2$ funciona mais rápido que um algoritmo cujo tempo de execução é 2^n na mesma máquina?

Valores de n	Algoritmo A ($100n^2$)	Algoritmo B (2^n)
1	100	2
5	2500	32
10	10.000	1.024
14	19600	16384
15	22500	32768

Para n=14

5. Considere dois programas A e B com complexidade $100n^2$ e $5n^3$, respectivamente. Qual é o mais eficiente?

Valor de n	$100n^2$	$5n^3$
1	100	5
...		
10	10.000	5.000
...		
20	40.000	40.000
21	44.100	46.305

Para n <20 o algoritmo B é mais eficiente
Para n=20 a eficiência é igual
Para n >20 A é mais eficiente.

6. Dois algoritmos A e B possuem complexidades n^5 e 2^n respectivamente. Você utilizaria o algoritmo B ao invés do A em qual caso. Exemplifique.

Valores de n	n^5	2^n
1	1	2
2	32	4
10	100000	1024
20	3.200.000	1.048.576
22	5.153.632	4.194.304
23	6.436.343	8.388.608

Sim. Para n>=2 e n<=22

7. Um algoritmo tem complexidade $2n^3$. Num certo computador **A**, em um tempo *t*, o algoritmo resolve um problema de tamanho 27. Imagine agora que você tem disponível um computador **B** 729 vezes mais rápido. Qual é o tamanho máximo do problema que o mesmo algoritmo resolve no mesmo tempo *t*?

$C = 2n^3$

T = C/Vc, nesse caso Ta = Tb e Vcb = 729Vca
Em A, n = 27 ? em B quanto será n ??
Logo...

$2(27)^3/Vca = 2n^3/729Vca$

$n^3 = 729(27)^3$
n= 27x9 = 243

Comparando as complexidades $2n^2$, $2n^3$ e 2^n ,

Complexidade	n	Velocidade Computador	Velocidade Computador	n
$2n^2$	25	Va	100Va	250
$2n^3$	27	Va	729Va	243
2^n	25	Va	125Va	32

E se for linear, ou seja C=n e Vb=100Va, e em Va n=25 Respondam