UENF

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Curso:	Ciência de Computação e Informática	**Data:**	26./.09./2022
Lista:	Questões sobre Complexidade	**Período:**	4º
Disciplina:	Estrutura de Dados II		
Professor:	Fermín A. Tang	**Turno:**	Diurno

Complexidade

1. Determine o O(.) para cada uma das seguintes funções que representam o número de passo requeridos por um dado algoritmo:

   (a) $T(n) = n^2 + 400n + 5$
   (b) $T(n) = 67n + 3n$
   (c) $T(n) = 2n + 5n \log n + 100$
   (d) $T(n) = \log n + 2n^2 + 55$
   (e) $T(n) = 3(2^n) + n^8 + 1024$
   (f) $T(n, k) = kn + \log k$
   (g) $T(n, k) = 9n + k \log n + 1000$

   Resposta.-

   (a) T(n) é O(n^2)
   (b) T(n) é O(n)
   (c) T(n) é O($n \log n$)
   (d) T(n) é O(n^2)
   (e) T(n) é O(2^n)
   (f) T(n, k) é O(kn)
   (g) T(n, k) é O($k \log n$)

2. Considere um problema P que pode ser resolvido usando diferentes algoritmos. A complexidade de cada algoritmo pode ser representada mediante uma função $f(n)$, onde n representa o tamanho do problema. Para cada função $f(n)$, determine o maior tamanho n de um problema que pode ser resolvido em tempo t conforme indicado na tabela. Considere que cada algoritmo leva $f(n)$ microssegundos para resolver um problema.

	1 Second	1 Hour	1 Month	1 Century
$\log n$	$\approx 10^{300000}$			
n				
$n \log n$				
n^2				
2^n				

   Resposta.- Considerando que o algoritmo linear n leva n microsegundos. Sabemos que um microsegundo equivale a 10^{-6} segundos.

   Para uma entrada n, o algoritmo linear leva n microsegundos:

   Em um segundo, ele conseguiria resolver um problema de tamanho:

$n = 10^6$ onde $10^6 \times 10^{-6}$ = 1 segundo

Em uma hora = 3.600 segundos ($3{,}6 \times 10^3$), ele conseguiria resolver um problema de tamanho:

$n = 3{,}6 \times 10^9$ onde $3{,}6 \times 10^9 \times 10^{-6} \leq 3.600$ segundos

Em um mês = ($30 \times 24 \times 3.600$) segundos ($2{,}592 \times 10^6$), ele conseguiria resolver um problema de tamanho:

$n = 2{,}592 \times 10^{12}$ onde $2{,}592 \times 10^{12} \times 10^{-6} \leq 2{,}592 \times 10^6$ segundos

Para uma entrada n, o algoritmo quadrático leva n^2 microsegundos:

Em um segundo, ele conseguiria resolver um problema de tamanho:

$n = 10^3$ onde $(10^3)^2 \times 10^{-6}$ = 1 segundo

Em uma hora = 3.600 segundos ($3{,}6 \times 10^3$), ele conseguiria resolver um problema de tamanho:

$n = 6 \times 10^4$ onde $(6 \times 10^4)^2 \times 10^{-6} \leq 3.600$ segundos

Para uma entrada n, o algoritmo quadrático leva 2^n microsegundos:

Em um segundo, ele conseguiria resolver um problema de tamanho:

$n = \log 10^6 = 19{,}93$ onde $2^n \times 10^{-6}$ = 1 segundo

Para uma entrada n, o algoritmo exponencial conseguiria:

$2^n = 10^6$ $n = \log 10^6$

	Tamanho do problema			
	1 segundo	1 hora	1 mês	1 século
$\log n$				
n	10^6	$3{,}6 \times 10^9$	$2{,}592 \times 10^{12}$	$3{,}1104 \times 10^{15}$
$n \log n$				
n^2	10^3	6×10^4	$1{,}609 \times 10^6$	$5{,}577 \times 10^7$
2^n	19,93	31,745	41,237	51,466

3. O número de operações realizadas pelos algoritmos A e B é $40n^2$ e $2n^3$ respectivamente. Determine o tamanho de n_0 de maneira que A seja melhor que B para $n \geq n_0$.
4. Os programas A e B foram analisados concluindo-se que os tempos de execução de pior caso não são maiores do que $150nlogn$ e n^2, respectivamente. Responda as seguintes questões:
   a) Qual programa oferece o melhor desempenho no tempo de execução para valores grandes de n (n>10.000)?

b) Qual programa oferece o melhor desempenho no tempo de execução para valores pequenos de n (n<100)?
c) Qual programa executara mais rápido em média para n=1.000.
d) Pode o programa B executar mais rápido que o programa A para todas as entradas possíveis?.

5. Assumindo que $f_1(n)$ é $O(g_1(n))$ e $f_2(n)$ é $O(g_2(n))$ prove:

a). $f_1(n) + f_2(n)$ é $O(\max(g_1(n), g_2(n))$

b) $f_1(n) * f_2(n)$ é $O(g_1(n) * g_2(n))$

Resposta.-

a) Dado que $f_1(n)$ é $O(g_1(n))$ e $f_2(n)$ é $O(g_2(n))$ sabe-se que:

$f_1(n) \leq c_1 g_1(n)$ para $n \geq n_1$

$f_2(n) \leq c_2 g_2(n)$ para $n \geq n_2$

Considerando $n \geq n_0 = \max(n_1, n_2)$podemos somar as duas expressões acima:

$$f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \quad \text{para } n \geq n_0$$

Considerando $c = \max(c_1, c_2)$ temos que:

$$f_1(n) + f_2(n) \leq c\big(g_1(n) + g_2(n)\big) \quad \text{para } n \geq n_0$$

Sabe-se que: $\big(g_1(n) + g_2(n)\big)/2 \leq max\big(g_1(n), g_2(n)\big) \leq \big(g_1(n) + g_2(n)\big)$

$$f_1(n) + f_2(n) \leq 2c\ max\big(g_1(n), g_2(n)\big) \quad \text{para } n \geq n_0$$

Fazendo $c_0 = 2c$ temos a prova desejada:

Assim provamos que: $f_1(n) + f_2(n)$ é $O(\max(g_1(n), g_2(n))$

b) Dado que $f_1(n)$ é $O(g_1(n))$ e $f_2(n)$ é $O(g_2(n))$ sabe-se que:

$f_1(n) \leq c_1 g_1(n)$ para $n \geq n_1$

$f_2(n) \leq c_2 g_2(n)$ para $n \geq n_2$

Considerando $n \geq n_0 = \max(n_1, n_2)$podemos multiplicar as duas expressões acima:

$$f_1(n) * f_2(n) \leq c_1 c_2 g_1(n) * g_2(n) \quad \text{para } n \geq n_0$$

Fazendo $c = c_1 c_2$ temos a prova desejada:

$$f_1(n) * f_2(n) \leq c g_1(n) * g_2(n) \quad \text{para } n \geq n_0$$

Assim provamos que: $f_1(n) * f_2(n)$ é $O(g_1(n) * g_2(n))$

6. Usando a definição de O(.) prove que:

a) 2^{n+a} é $O(2^n)$
b) 2^n é $O(n!)$ e $n!$ não é $O(2^n)$
c) $\sum_{i=1}^{n} i^2$ é $O(n^3)$ e em geral $\sum_{i=1}^{n} i^k$ é $O(n^{k+1})$

Resposta.-

a) Podemos expressar: $2^{n+a} = 2^n * 2^a$

Fazendo $c = 2^a$ temos que: $2^{n+a} \leq c2^n$ para $n \geq 0$

Temos assim que: 2^{n+a} é $O(2^n)$

b) Sabe-se que: $2^n = 2 \times 2 \times 2 \times ... \times 2 \leq 2 \times (1 \times 2 \times 3 \times ... \times n)$

Com isso temos que: $2^n \leq cn!$ para $c = 2$ e $n \geq 1$

Assim provamos que: 2^n é $O(n!)$

c) Sabe-se que: $\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 \leq n^2 + n^2 + \cdots + n^2 = n^3$

Com isso temos que: $\sum_{i=1}^{n} i^2 \leq cn^3$ para $c = 1$ e $n \geq 1$

Assim provamos que: $\sum_{i=1}^{n} i^2$ é $O(n^3)$

De maneira análoga, temos que:

$\sum_{i=1}^{n} i^k = 1^k + 2^k + \cdots + n^k \leq n^k + n^k + \cdots + n^k = n.n^k = n^{k+1}$

Com isso temos que: $\sum_{i=1}^{n} i^k \leq cn^{k+1}$ para $c = 1$ e $n \geq 1$

Assim provamos que: $\sum_{i=1}^{n} i^k$ é $O(n^{k+1})$