$T = C \cdot ti$ / $aT\left(\frac{n}{b}\right) + (n^d)$

# BCC241 – Projeto e Análise de Algoritmos
## Prova I – Valor 10 – peso 2

decom
departamento de computação

UFOP

Data: 05/08/2024
Professor: Anderson Almeida Ferreira

Nome: Felipe Braz Marques Matrícula: 22.1.4030

1. (1,5 ponto) Para cada função f(n) a seguir, indique e justifique se ela é O(g(n)), o(g(n)), Θ( g(n)), ω(g(n)) e Ω(g(n)) (informe a justificativa para cada notação assintótica):

   a. $f(n) = 5n + 5$ , $g(n) = 4n^2 + 10$

   b. $f(n) = n^2$ , $g(n) = n \log n$

2. (1,5 ponto) Para cada função T(n) de complexidade de tempo a seguir, determine o maior tamanho $\boxed{n}$ de um problema que pode ser resolvido em 1 minuto. Considere que o tempo para dada instrução é $10^{-6}$ segundos.

   a) $T(n) = 4^n$

   b) $T(n) = n^4$

3. (1,5 ponto) Para cada item a seguir, responda V se o item for verdadeiro e F caso seja falso e, para cada situação, justifique a sua resposta.

   (   ) Se o pior caso do tempo de computação de um algoritmo A é $\Omega(n^2)$, então é possível que este algoritmo seja $O(n^2)$ no melhor caso.

   (   ) $n^2 = \Omega(2n^4)$

   (   ) $n \log n = \Theta(n \log n^2)$

4. (2 pontos) Para o trecho de código a seguir, forneça a sua **função de complexidade** de tempo no melhor caso, pior caso e caso médio. Considere cada instrução básica com custo 1 e que a probabilidade da condição do comando *if* ser verdadeira ou falsa são iguais para o caso médio (Não é para fornecer o resultado usando notação assintótica).

```
L1 a=0;
L2 if (x==y) {
    L3 for (i = 0; i < N; i++) {
        L4 a = a + 1;
        L5 a = a * 2;
    }
}
L6 else {
    L7 for (i = 0; i < N; i++) {
        L8 for (j = 0; j < N; j++) {
            L9 a = a + 2;
        }
    }
}
```

5. (2 pontos) A função a seguir realiza a multiplicação de dois **números inteiros grandes**, $x$ e $y$, retornando o produto. $x$ e $y$ possuem $n$ bits.
Encontre o limite assintótico superior **justo** desta função, no pior caso, analisando cada linha do algoritmo. O tamanho da entrada corresponde a quantidade de bits para representar os números.

**Figure 1.1** Multiplication à la Français.

```
function multiply(x, y)
Input:  Two n-bit integers x and y, where y ≥ 0
Output:  Their product

L1 if y = 0:  return 0
L2 z = multiply(x, ⌊y/2⌋)
L3 if y is even:
  L4 return 2z
  else:
  L5 return x + 2z
```

6. (1,5 ponto) Ao desenvolver um algoritmo, usando divisão e conquista, para tratar um determinado problema inicial de tamanho *n*, o programador, em seu código, divide o problema inicial de tamanho *n* em *2* subproblemas de tamanho *n/4* a um custo *O(1), resolve recursivamente os 4 subproblemas* e combina os resultados das resoluções dos subproblemas a um custo *O(n)*. Explique (justificando) o quê o programador deve melhorar no algoritmo para diminuir a ordem de complexidade do mesmo.

--------------------------------

Alguns somatórios:

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1} \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Soma dos termos de uma p.a. finita: $S_n = a_1 + a_2 + a_3 + \ldots + a_{n-2} + a_{n-1} + a_n = n(a_1 + a_n)/2$

Soma dos termos de uma p.g. finita: $S_n = a_1 + a_1 q + a_1 q^2 + \ldots + a_1 q^{n-2} + a_1 q^{n-1} = a_1 (q^n - 1)/(q-1)$

Felipe Braz Marques / 22.1.4030

8,3

1) a) $f(n) = 5n+5$, $g(n) = 4n^2+10$

$$\lim_{n \to \infty} \frac{5n+5}{4n^2+10} \Rightarrow \lim_{n \to \infty} \frac{\overbrace{5n/n^2}^{\text{Tende}} + \overbrace{5/n^2}^{\text{Tende a 0}}}{\underbrace{4n^2/n^2}_{\to 4} + \underbrace{10/n^2}_{\text{Tende a 0}}} \Rightarrow 0$$ ?

E as outras notações?

Como limite tende a 0, a função é $o(g(n))$ e $O(g(n))$.

1,0

b) $f(n) = n^2$, $g(n) = n \log n$

$$\lim_{n \to \infty} \frac{n^2}{n \log n} \Rightarrow \lim_{n \to \infty} \frac{\cancel{n} \cdot n}{\cancel{n} \log n} \Rightarrow \lim_{n \to \infty} \frac{n}{\log n} = \infty$$

Como limite tende a $\infty$, a função é $\omega(g(n))$ e $\Omega(g(n))$.

2) $T = c \cdot ti$

$C = \frac{T}{ti}$

a) $T(n) = 4^n$

$4^n = \frac{60}{10^{-6}}$

$2^{2n} = 60 \cdot 10^6$

$\lg 2^{2n} = \lg 60 \cdot 10^6$

$2n = \lg 60 \cdot 10^6$

$n = \frac{\lg 60 \cdot 10^6}{2}$

1,5

b) $T(n) = n^4$

$n^4 = \frac{60}{10^{-6}}$

$n^4 = 60 \cdot 10^6$

$n = \sqrt[4]{60 \cdot 10^6}$

3) (V) É verdadeiro pois pelo cálculo de limite se deu uma constante diferente de 0, a função é $\Omega(n^2)$, $O(n^2)$ e $\Theta(n^2)$.

1,3

(F) Como o limite deu 0, a função é $O(2n^4)$ e $O(2n^4)$.

$$\lim_{n\to\infty} \frac{n^2}{2n^4} \Rightarrow \lim_{n\to\infty} \frac{\cancel{n^2}/n^4}{2\cancel{n^4}/\cancel{n^4}} \Rightarrow \lim_{n\to\infty} \frac{1/n^{2}}{2\cdot 1} \xrightarrow{\text{tendendo } 0} = 0$$

(V) Como o limite deu 1/2, a função é $\Omega(n \log n^2)$, $O(n \log n^2)$ e $\Theta(n \log n^2)$.

$$\lim_{n\to\infty} \frac{n \log n}{n \log n^2} \Rightarrow \lim_{n\to\infty} \frac{\cancel{n \log n}}{2\,\cancel{n \log n}} \Rightarrow \frac{1}{2}$$

4)

| Linhas | instruções | repetições |
|---|---|---|
| L1 | 1 | 1 |
| L2 | 1 | 1 |
| L3 | 2 | $n+1$ |
| L4 | 1 | $n$ |
| L5 | 1 | $n$ |
| L7 | 2 | $n+1$ |
| L8 | 2 | $n.(n+1)$ |
| L9 | 1 | $n^2$ |

• Melhor caso entra no if:

$1+1+2(n+1)+n+n$

$2n+2+1+1+n+n$

$4n+4$ //

2,0

• Pior caso entra no else:

$1+1+2(n+1)+2(n^2+n)+n^2$

$1+1+2n+2+2n^2+2n+n^2$

$3n^2+4n+4$ //

• Caso médio:

$$\frac{3n^2+4n+4+4n+4}{2}$$

$$\frac{3n^2+8n+4}{2} \quad //$$

5) $L1 = O(n)$ pois verifica se não é 0, olhando todos os bits.
$L2 = O(n)$ shift
$L3 = O(1)$ olha o bit mais significativo
$L4 = O(n)$ shift
$L5 = O(n) + O(n) = O(n)$ soma e shift

2,0

$O(n) + O(n) + O(1) + O(n) + O(n) = O(n)$

Porém o pior caso a a entrada $N$ possui o bit mais a esquerda possível, fazendo com que $N = n$, o que fica $n \cdot O(n) = O(n^2)$.

6) $T(n) = aT\left(\frac{n}{b}\right) + (n^d)$

$4T\left(\frac{n}{2}\right) + (n^1)$

$\log_b a \quad d$
$\log_2 4 \quad 1$
$2 > 1$

0,5

Pelo teorema mestre temos que $O(n^2)$. Para melhorar o desempenho do seu algoritmo esse programador ~~precisa~~ aumentar a quantidade de de ~~subproblemas~~.