UNIVERSIDADE FEDERAL DE SÃOCARLOS
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação

Projeto e Análise de Algorítmos – AA1
Análise Assintótica

Prof. Dr. Alexandre Levada

Dupla:

Leticia Bossatto Marchezi. RA 759392. leticiabossatto@estudante.ufscar.br.
Paula Vitoria Martins Larocca. RA 769705. paula.larocca@estudante.ufscar.br.

July 9, 2023
São Carlos, SP

Resolução – Atividade Avaliativa 1 – Análise Assintótica (Projeto e Análise de Algoritmos)

Novembro - 2022 / Leticia Bossatto Marchezi e Paula Martins Larocca

Questão 2

Defina o que é a notação Ω. Faça um gráfico ilustrativo para exemplificar sua definição.

Resolução:

A notação Ω é uma estimativa do custo computacional mínimo de um algoritmo. É um limitante inferior pois representa o melhor caso, onde há a menor quantidade de passos possíveis.

Tomando arbitrariamente um valor de n, o limite para valores superiores a n na função C g(n) sempre terá valor inferior a f(n).

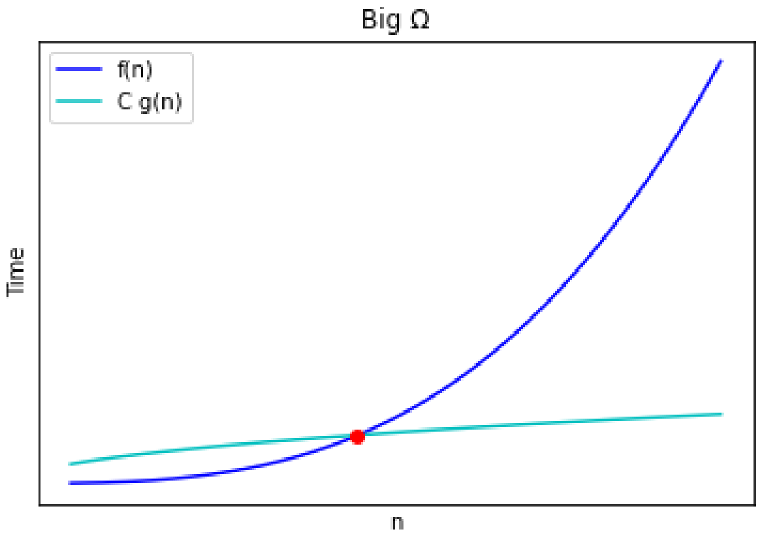

Figure 1: Função big Ω

Como observado na figura acima, ao tomar o intervalo a partir do ponto de interseção entre as duas funções pode-se dizer que a curva C g(n) é um limitante inferior a f(g). □

Questão 4

Defina a função T(n) que conta quantas operações serão executadas pelo script Python a seguir. Calcule as notações Big-O, Ω e Θ. Explique como você obteve sua resposta.

```
def Algo_C(n):
    a = 100
    j = n
    while j > 0:
        k = 0
        while k < j:
            a = a + 10
            k = k + 1
        j = j - 1
```

```
    return a
```

Resolução:

A função $Algo_C(n)$ possui dois laços de repetições, um externo em que haverá n repetições e outro interno com quantidade variável de iterações.

O laço externo (while j>0) executará n vezes, pois j toma o valor inicial de n e será decrementado até ter valor igual a 0. Já o laço interno(while k<j) executará j vezes em cada iteração do laço externo.

Dessa forma, para j = n, a variável a será incrementada n vezes no total. Na próxima iteração, para j = n-1, ocorrerão n-1 incrementos e assim por diante.

Assim, o número de passos do algoritmo pode ser representado pelo seguinte somatório:

$$n + (n-1) + (n-2) + .. + (n-x) \Rightarrow \sum_{j=n}^{1} k \tag{2.1}$$

para x $\in \mathcal{Z}$ e x $<$ n.

Pela propriedade comutativa da soma, temos que:

$$\sum_{j=n}^{1} k = \sum_{j=1}^{n} k \tag{2.2}$$

E veja que tem-se a seguinte igualdade

$$(k+1)^2 = k^2 + 2*k + 1 \Rightarrow (k+1)^2 - k^2 = 2*k+1 \tag{2.3}$$

Representando em somatórios:

$$\sum_{j=1}^{n}(k+1)^2 - k^2 = \sum_{j=1}^{n} 2*k+1 \tag{2.4}$$

$\sum_{j=1}^{n}(k+1)^2 - k^2$ é uma soma telescópica, logo, pode-se simplificar a expressão pela subtração do último termo com o primeiro:

$$(n+1)^2 - 1 = \sum_{j=1}^{n} 2*k+1 \tag{2.5}$$

Decompondo os somatórios em termos únicos:

$$(n+1)^2 - 1 = 2 * \sum_{j=1}^{n} k + \sum_{j=1}^{n} 1 \tag{2.6}$$

Desenvolvendo o binômio quadrado perfeito e isolando o somatório em k:

$$\sum_{j=1}^{n} k = \frac{(n^2 + 2*n + 1 - 1 - n)}{2} \tag{2.7}$$

Pode-se simplificar os termos para:

$$\sum_{j=1}^{n} k = \frac{(n^2 + n)}{2} \tag{2.8}$$

Dessa forma, a função T(n) que conta a quantidade de atribuições no algoritmo, considerando as 2 atribuições iniciais é:

$$T(n) = \frac{(n^2 + n)}{2} + 2 \tag{2.9}$$

Assim, analisando a taxa de crescimento de T(x), o termo dominante é n^2 . Resultando em:

$$\Theta(n^2), O(n^2), \Omega(n^2) \tag{2.10}$$

□

Questão 6

Mostre que se c é um número real positivo, então:

$$g(n) = \sum_{k=0}^{n} c^k \tag{3.1}$$

é:

(a) Θ(n) , se c = 1

Resolução:

Se c=1 tem-se que:

$$g(n) = \sum_{k=0}^{n} 1^k \tag{3.2}$$

Como o somatório é uma potência de base um, para qualquer valor de k no intervalo delimitado o termo terá valor um. Ou seja, o somatório é composto por n somas de 1.

Assim, pode-se concluir que o resultado do somatório é a quantidade de termos existentes nele, ou seja, n.

$$g(n) = \sum_{k=0}^{n} 1^k = n \tag{3.3}$$

Logo, T(n) é:

$$T(n) = n \tag{3.4}$$

Por consequência resultando em Θ (n).

□

(b) $\Theta(c^n)$, se c > 1

Resolução:

Se c>1 tem-se que:

$$g(n) = \sum_{k=0}^{n} c^k \tag{3.5}$$

Valendo a seguinte igualdade:

$$c^{k+1} = c^k * c => c^{k+1} = c^k + c^k => c^{k+1} - c^k = c^k \tag{3.6}$$

Assim, representando em termos de somatórios:

$$\sum_{k=0}^{n} c^k = \sum_{k=0}^{n} c^{k+1} - c^k \tag{3.7}$$

O termo à direita é uma soma telescópica, podendo ser simplificada para:

$$\sum_{k=0}^{n} c^k = (c^{n+1} - c^n) - (c^0) \tag{3.8}$$

$$T(n) = \sum_{k=0}^{n} c^k = c^n - 1 \tag{3.9}$$

Assim, o termo dominante em T(n) é c^n.

Logo o custo do algoritmo é Θ (c^n) e significa que seu custo.

□

(c) $\Theta(1)$, se c $<$ 1

Resolução:

Para c<1 tem-se uma função decrescente em que para valores maiores de n, o termo da somatória será menor. Ou seja, em um limite de n tendendo a infinito, os termos tendem a 0. Assim, o primeiro termo será o que dominará na soma.

Por isso, diz-se que o custo das operações é constante, pois o resultado da somatória não é significantemente alterado por valores discrepantes de n.

Logo, temos Θ (1).

□

Questão 8

Mostre que f (n)=n! é O(n) .

□

Resolução:

Tomando n como um número inteiro diferente de 0, tem-se que:

$$n! = n(n-1)! = n(n-1)(n-2)! \tag{4.1}$$

E assim por diante.

Dessa forma prova-se que:

$$n! < n^n \tag{4.2}$$

Como a notação Big O representa um limitante superior, a função fatorial está abaixo do limitante e não é uma aproximação distante, então conclui-se que n! é O(n^n).

Questão 10

Para cada uma das funções a seguir, indique se f=O(g) , f =$\Omega(g)$ ou f =Θ(g), justificando cada uma das respostas:

(a) f(n) = $n^{1/2}$ e g(n) = $n^{2/3}$

Resolução:

A análise assintótica entre duas funções pode ser realizada a partir do estudo do limite da divisão entre elas. Dessa forma, tem-se que:

$$\lim_{x\to\infty} \frac{f(n)}{g(n)} = \lim_{x\to\infty} \frac{n^{1/2}}{n^{2/3}} \tag{5.1}$$

Simplificando a expressão:

$$\lim_{x\to\infty} n^{1/2-2/3} = \lim_{x\to\infty} n^{3-4/6} = \lim_{x\to\infty} n^{-1/6} \tag{5.2}$$

Como $\lim_{x\to\infty} n^{-1/6}$ tende a infinito, pode-se concluir que f(n) cresce mais rapidamente do que g(n) e então f(n) = Ω (g)

□

(b) f(n) = $10 \log n$ e g(n) = $\log n^2$

Resolução:

É possível aplicar a propriedade do exponente no logaritmo e obter $\log n$ no numerador e denominador, anulando-os

$$\lim_{x\to\infty} \frac{10\log n}{\log n^2} = \lim_{x\to\infty} \frac{10\log n}{2 * \log n} = 10/2 = 5 \tag{5.3}$$

Já que $\lim_{x\to\infty} \frac{10\log n}{\log n^2}$ é 5, pode-se concluir que f(n) e g(n) possuem taxa de crescimento proporcionais, então f(n) = Θ (g)

□

(c) f(n) = $\sqrt{n}$ e g(n) = $(\log n)^3$

Resolução:

Como o limite tende a indefinição ∞/∞ é possível aplicar L'Hôpital sucessivamente:

$$\lim_{n\to\infty} \frac{\sqrt{n}}{(\log n)^3} = \lim_{x\to\infty} \frac{\frac{1}{2*n^{1/2}}}{\frac{3*(\log n)^2}{n}} = \lim_{n\to\infty} \frac{\log(10) * \sqrt{n}}{6 * (\log n)^2} \tag{5.4}$$

$$\lim_{n\to\infty} \frac{\log(10) * \sqrt{n}}{6 * (\log n)^2} = \infty \tag{5.5}$$

Como $\lim_{n\to\infty} \frac{\sqrt{n}}{(\log n)^3}$ tende a infinito, pode-se concluir que f(n) cresce mais rapidamente do que g(n) e então f(n) = Ω (g)

□

(d) f(g) = $n^{0.01}$ e g(n) = $\log n$

Resolução:

$$\lim_{n\to\infty} \frac{n^{1/100}}{\log n} = \lim_{n\to\infty} \frac{n}{100 * n^{99/100}} = \lim_{n\to\infty} \frac{n}{100 * n^{99/100}} = \lim_{n\to\infty} \frac{n^{1/100}}{100} = \infty \tag{5.6}$$

Como $\lim_{n\to\infty} \frac{\sqrt{n}}{(\log n)^3}$ tende a infinito, pode-se concluir que f(n) cresce mais rapidamente do que g(n) e então f(n) = Ω (g)

□