Lista 1

18:39 19:48
19:16

1- • Complexidade de tempo: Indica a rapidez com que um determinado algoritmo é executado.

• Complexidade de espaço: Indica a quantidade de unidades de memória utilizadas pelo algoritmo

* Em geral nos concentramos na complexidade de tempo.

↳ Como medir a complexidade de tempo?

↳ "Contando" o número de operações básicas realizadas em função do tamanho da entrada

* Em geral, estamos mais interessados na ordem de crescimento do que no número exato de operações.

• Operação básica: É a operação que mais contribui para o tempo de execução do algoritmo. É a operação mais custosa que é executada pelo maior número de vezes.

$T(n) \approx C_{op} C(n)$

$T(n)$: Tempo de execução
$C(n)$: ↳ Complexidade de tempo
C_{op}: ↳ Custo da operação básica.

Exemplo:

$$C(n) = \frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \approx \frac{1}{2}n^2$$

$$\frac{T(2n)}{T(n)} \approx \frac{C_{op}((2n)}{C_{op}C(n)} = \frac{\frac{1}{2}(2n)^2}{\frac{1}{2}n^2} = 4$$

2- Pior caso: É a situação para a qual o algoritmo executa o maior número de operações para uma entrada de tamanho n.

Eficiência de pior caso: É a eficiência do algoritmo para a entrada de tamanho n que faz com que ele execute o maior número de operações.

* Análise geralmente feita. Forma de estabelecer limites superiores para a execução de um algoritmo

Eficiência do melhor caso: É a eficiência para a entrada de tamanho n que faz com que o algoritmo execute o menor número de operações possível

* Não é tão utilizada mas pode ser útil conhecer quais são os melhores casos de um algoritmo. Comentar insertion Sort

Eficiência de caso médio: É o desempenho esperado de um algoritmo. Mede o número de operações considerando uma distribuição de probabilidades sobre todas as possíveis entradas.

* Bem mais difícil de calcular mas fornece a visão mais realista do desempenho de um algoritmo.

Exemplo: Sequential Search $(A[0 \ldots n-1], k)$

```
i ← 0
while i < n and A[i] ≠ k do
    i ← i + 1
if i < n
    return i
else
    return -1
```

operação básica: adição (+)

Pior caso: K não está no array A

$$T(n) = \sum_{i=0}^{n-1} 1 = \underbrace{1 + 1 + \cdots + 1}_{?}$$

$$= \left(\left((n-1) - 0\right) + 1\right) \cdot 1 = n$$

Melhor caso: K está em $A[0]$

$$T(n) = 1$$

Caso médio:

$$T(n) = \sum_{i=0}^{n-1} (\text{comparações para a posição } i \times \text{probabilidade de estar na posição } i)$$

$$= \sum_{i=0}^{n-1} \left(i \times \frac{1}{n}\right)$$

$$= \frac{1}{n} \sum_{i=0}^{n-1} i$$

$$= \frac{1}{n} \left(\frac{n(n+1)}{2}\right) = \frac{n+1}{2}$$

4- a) $\log_2 n$

$\log_2 4n - \log_2 n$

$= (\log_2 4 + \log_2 n) - \log_2 n$

$= 2$ operações

b) $\frac{\sqrt{4n}}{\sqrt{n}} = \frac{2\sqrt{n}}{\sqrt{n}} = 2\times$ mais operações

c) $\frac{4n}{n} = 4$

d) $\frac{(4n)^2}{n^2} = \frac{16n^2}{n^2} = 16$

e) $\frac{(4n)^3}{n^3} = 64$

f) $\frac{2^{4n}}{2^n} = 2^{4n-n} = 2^{3n}$

5 - a) $\frac{\sqrt[3]{(10^3)^3}}{\sqrt[3]{(5\cdot 10^{-2})^3}} = \frac{(10^3)^3}{5^3 \cdot 10^6} = \frac{10^9}{5^3 \cdot 10^6} = \frac{10^3}{5^3} = \frac{2^3 \cdot 5^3}{5^3} = 8$

b) Para um problema de tamanho n gastamos + tempo para executar com $\frac{1}{3}n^3$ operações.

$t_{old} = C_M \frac{1}{3} n^3$

$t_{novo} = 10^{-3} C_M \frac{1}{3} N^3$

$t_{old} = t_{novo}$

$C_M \frac{1}{3} n^3 = 10^{-3} C_M \frac{1}{3} N^3$

$n^3 = 10^{-3} N^3$

$n = \frac{N}{10}$

6 - O - notation

$t(n)$ está em $O(g(n))$, $t(n) \in O(g(n))$, se $t(n)$ for limitada superiormente por algum multiplo constante de $g(n)$ para todos valores grandes de n, ou seja:

$t(n) \leq c g(n)$ para todo $n \geq n_0$

$c > 0$, $n_0 \geq 0$

Ω - notation

$t(n) \geq c g(n)$ para todo $n \geq n_0$

$c > 0$, $n_0 \geq 0$

Θ - notation

$c_2 g(n) \leq t(n) \leq c_1 g(n)$ para todo $n \geq n_0$

$c_1, c_2 > 0$, $n_0 \geq 0$

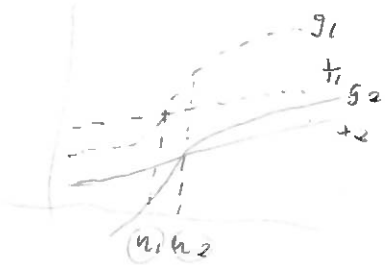
7 -

Teorema: Se $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$

então $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

$t_1(n) \leq c\, g_1(n) \quad n \geq n_1$

$t_2(n) \leq b\, g_2(n) \quad n \geq n_2$



Propriedade:

- $a_1, b_1, a_2, b_2 \in \mathbb{R}$

Se

$a_1 \leq b_1$ e $a_2 \leq b_2 \Rightarrow a_1 + a_2 \leq 2\max\{b_1, b_2\}$

$\Rightarrow t_1(n) + t_2(n) \leq a\, g_1(n) + b\, g_2(n), \quad n \geq \max(n_1, n_2)$

$d = \max(a, b)$

$\Rightarrow t_1(n) + t_2(n) \leq d\, g_1(n) + d\, g_2(n)$

$g_3 = \max(g_1(n), g_2(n))$

$\Rightarrow t_1(n) + t_2(n) \leq 2d\, g_3(n), \quad n \geq \max(n_1, n_2)$

$\Rightarrow t_1(n) + t_2(n) \in O(g_3(n)), \quad c = 2d$

$= 2\max(a, b)$

$= t_1(n) + t_2(n) \in O(\max(g_1(n), g_2(n))) \quad n_0 = \max(n_1, n_2)$

8.-

a) $$\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} = \frac{1}{2}\lim_{n\to\infty} \frac{n^2-n}{n^2} = \frac{1}{2}\lim_{n\to\infty}\left(1-\frac{1}{n}\right) = \frac{1}{2}$$

b) $$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 n)'}{(\sqrt{n})'} = \lim_{n\to\infty} \frac{(\log_2 e)\frac{1}{n}}{\frac{1}{2\sqrt{n}}}$$

$$= 2\log_2 e \lim_{n\to\infty}\left(\frac{\sqrt{n}}{n}\cdot\frac{\sqrt{n}}{\sqrt{n}}\right) = 2\log_2 e \lim_{n\to\infty}\left(\frac{n}{n\sqrt{n}}\right) = 0$$

c) $$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}$$

$$= \lim_{n\to\infty} \sqrt{2\pi n}\,\frac{n^n}{2^n e^n} =$$

$$= \lim_{n\to\infty} \sqrt{2\pi n}\left(\frac{n}{2e}\right)^n = \infty$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0, & f(n) \text{ tem ordem de crescimento menor que } g(n) \\ c, & \text{mesma ordem de crescimento} \\ \infty, & f(n) \text{ tem ordem de crescimento maior.} \end{cases}$$

10 -

$$\lim_{n\to\infty} \frac{p(n)}{n^k} = \lim_{n\to\infty} \left(\frac{a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} + \dots + a_0}{n^k} \right)$$

$$= \lim_{n\to\infty} \left(a_k + \frac{a_{k-1}}{n} + \frac{a_{k-2}}{n^2} + \dots + \frac{a_0}{n^k} \right) = a_k$$

Para o que serve esse resultado.

11 -

$$\lim_{n\to\infty} \left(\frac{a_1^n}{a_2^n} \right) = \lim_{n\to\infty} \left(\frac{a_1}{a_2} \right)^n$$

- Se $a_1 > a_2 \to \infty$
- Se $a_1 < a_2 \to 0$
- Se $a_1 = a_2 \to 1$

$2^n \in O(3^n)$

$(10n)^2 \in \Theta(n^2)$

$(5n)^2 \in \Theta(n^2)$

12 -

Algoritmo Mystery(n):

```
S ← 0
for i ← 1 to n do
    S ← S + i * i
return S
```

a) $\sum_{i=0}^{n} i^2$ → Soma dos quadrados dos números inteiros de 1 a n

b) Multiplicação ou adição

c) $T(n) = \sum_{i=1}^{n} 1 = (n - 1 + 1)\,1 = n$

Classe de complexidade $\Theta(n)$

d) $\Theta(n)$

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

Nota:

$$\sum_{i=a}^{b} c a_i = c \sum_{i=a}^{b} a_i$$

$$\sum_{i=k}^{p} (a_i \pm b_i) = \sum_{i=k}^{p} a_i \pm \sum_{i=k}^{p} b_i$$

$$\sum_{i=k}^{p} 1 = p - k + 1$$

$$\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=k}^{p} i = \frac{(k+p)(p-k+1)}{2}$$

```
Secret (A[0,...,n-1])
    minval ← A[0] ; maxval ← A[0]
    for i ← 1 to n-1 do
        if A[i] to n-1 do
            if A[i] < minval
                minval ← A[i]
            if A[i] > maxval
                maxval ← A[i]
    return maxval - minval
```

a) Computa a amplitude

b) Comparação

c) $$T(n) = \sum_{i=1}^{n-1} 2 = 2\left((n-1)-1+1\right) = 2(n-1)$$

d) $\Theta(n)$

e) Usa... else if

```
Enigma (A[0... n-1, 0... n-1])
    for i ← 0 to n-2 do
        for j ← i+1 to n-1 do
            if A[i,j] ≠ A[j,i]
                return false
```

a) Verifica se a matriz é simétrica

b) Comparação

c) $$T(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} \left((n-1)-(i+1)+1\right)$$
$$= \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2}$$

d) $\Theta(n^2)$

13 -

a) $T(n) = T(n-1) + 5 \quad , \quad n > 1 \quad , \quad T(1) = 0$

$= (T(n-2) + 5) + 5$

$= ((T(n-3) + 5) + 5) + 5$

$\vdots$

$= T(n-i) + 5i$

$\vdots \quad i = n-1$

$= T(n-(n-1)) + 5(n-1)$

$= T(1) + 5n - 5 \in \Theta(n)$

b) $T(n) = 3T(n-1) \quad , \quad n > 1 \quad , \quad T(1) = 4$

$= 3(3T(n-2))$

$= 3(3(3T(n-3)))$

$\vdots$

$= 3^i T(n-i)$

$\vdots \quad i = n-1$

$= 3^{n-1} T(1) = 4 \cdot \frac{1}{3} 3^n \in O(3^n)$

c) $T(n) = T(n-1) + n \quad , \quad n > 0 \quad , \quad T(0) = 0$

$= (T(n-2) + n) + n$

$= ((T(n-3) + n) + n) + n$

$\vdots$

$= T(n-i) + in$

$\vdots \quad i = n$

$= T(0) + n^2 = n^2 \in \Theta(n^2)$

d) $T(n) = T(n/2) + n \quad , \quad n > 1 \quad , \quad T(1) = 1$

para $n = 2^k$

$T(2^k) = T(2^{k-1}) + 2^k$

$= (T(2^{k-2}) + 2^{k-1}) + 2^k$

$= ((T(2^{k-3}) + 2^{k-2}) + 2^{k-1}) + 2^k$

$\vdots$

$= T(2^{k-i}) + 2^{k-(i-1)} + 2^{k-(i-2)} + \ldots + 2^k$

$\vdots \quad i = k$

$= T(2^0) + 2^1 + 2^2 + \ldots + 2^k$

$= T(1) + \sum_{i=1}^{k} 2^i = \sum_{i=0}^{k} 2^i = \frac{1 - 2^{k+1}}{1 - 2} = \frac{1 - 2n}{-1}$

$= 2n - 1 \in O(n)$

$$S_n = \sum_{k=0}^{n} ar^k = \frac{a(1 - r^{n+1})}{(1 - r)}$$

$$S_n = ar^0 + ar^1 + \cdots + ar^n$$

$$rS_n = ar^1 + ar^2 + \cdots + ar^{n+1}$$

$$S_n - rS_n = ar^0 - ar^{n+1}$$

$$S_n(1 - r) = a(r^0 - r^{n+1})$$

$$S_n = \frac{a(1 - r^{n+1})}{1 - r}$$

e) $T(n) = T(n/3) + 1$, $n > 1$, $T(1) = 1$

para $n = 3^k$

$$T(3^k) = T(3^{k-1}) + 1$$

$$= (T(3^{k-2}) + 1) + 1$$

$$= ((T(3^{k-3}) + 1) + 1) + 1$$

$$\vdots$$

$$= T(3^{k-i}) + i$$

$$\vdots \quad i = k$$

$$= T(3^0) + k = k = \log_3 n \in O(\log_3 n)$$

$$3^k = n$$

$$k \log_3 3 = \log_3 n$$

$$\frac{d}{dx} \log_a x = \frac{1}{x \ln(a)}$$

$$\lim_{n \to \infty} \frac{\log_3 n}{\log_2 n} = \lim_{n \to \infty} \frac{\frac{1}{\ln(3)\, n}}{\frac{1}{\ln(2)\, n}} =$$

$$= \lim_{n \to \infty} \frac{\ln(2)\, n}{\ln(3)\, n} = \frac{\ln(2)}{\ln(3)}$$

$$\left(\log_x n \in O(\log_y n) \right)$$

14- a)

```
def two-power-n(n)
    if (n == 0):
        return 1
    else:
        return two-power-n(n-1) + two-power-n(n-1)
```

b) $T(n) = 2T(n-1) + 1 \quad , \quad T(0) = 0$

$$= 2(2T(n-2)+1) + 1$$

$$= 2(2(2T(n-3)+1)+1)+1$$

$$= 2^3 T(n-3) + \overset{2^2}{4} + \overset{2^1}{2} + \overset{2^0}{1}$$

$$\vdots$$

$$= 2^i T(n-i) + \sum_{j=0}^{i-1} 2^j$$

$$\vdots \quad i = n$$

$$= 2^n T(n-n) + \sum_{j=0}^{n-1} 2^j = \frac{(1 - 2^{(n-1)+1})}{1-2} = 2^n - 1$$

$$\in O(2^n)$$

15 -

```
Algorithm Riddle (A[0, ... , n-1])
    if n == 1:
        return A[0]
    else: temp ← Riddle(A[0 ... n-2]):
        if temp ≤ A[n-1]
            return temp
        else
            return A[n-1]
```

a) Retorna o menor elemento do array

b) Operação básica : Comparação

$$T(n) = T(n-1) + 2 \quad , \quad T(1) = 1$$

$$= (T(n-2) + 2) + 2$$

$$= ((T(n-3) + 2) + 2) + 2)$$

$$\vdots$$

$$= T(n-i) + 2i$$

$$\vdots \; i = n-1$$

$$= T(1) + 2(n-1) = 2n-1$$

16 - Operação básica : Comparação

Quantas vezes a operação básica é executada:

$$T(n) = T(n-1) + 1 + \sum_{i=0}^{n-2} 1 \quad , \text{ para } n > 1 \; , \; T(1) = 1$$

$$= T(n-1) + n$$

$$= (T(n-2) + n) + n$$

$$\vdots$$

$$= T(n-i) + in$$

$$\vdots i = n-1$$

$$= T(1) + (n-1)n = n^2 - n \in \Theta(n^2)$$