Projeto e analíse de algoritmos

Amanda Goulart

Novembro 2021

1 Sejam as funções f(n) = (n + 1)² e g(n) = n² . Demonstre, aplicando as definições de notação assintótica, que as seguintes afirmações são verdadeiras ou mostre que são falsas.

1.1 f(n) = O(g(n))

Sabendo que a Notação O(g(n)) = f(n) : existe constantes positivas C1 e N0, tais que $0 \leq f(n) \leq c * g(n), \forall n \geq n0$

Tendo f(n) = (n-1)² e g(n) = n²:

Supondo : $f(n) = (n-1) \in O(n)$, teremos que

$0 \leq (n-1) \leq c * n$

$0 \leq \frac{n^2-2n+1}{n^2} \leq c$

Adotando um N0=2, pois é necessário ser N0 > que 0.

$0 \leq \frac{2^2-2*2+1}{2^2} \leq c$

$0 \leq \frac{1}{4} \leq c$

Agora, fazendo a analise para ∞ :

$\lim_{x\to\infty} \frac{n^2-2n+1}{n^2} \quad = 1$

$\frac{1}{4} - 1 = -\frac{3}{4}$

$c = -\frac{3}{4}$

n0 = 2

$0 \leq \frac{n^2-2n+1}{n^2} \leq -\frac{3}{4} * n, \forall n \geq 2$

Note que esta ultima afirmação é falsa, portanto f(n) != O(g(n)).

1.2 f(n) = Ω(g(n))

Sabendo que a notação Ω (g(n)) = f(n) : existem constantes positivas c e n0 | $0 \leq c * g(n) \leq f(n) \forall n \geq n0$

f(n) = (n-1)² ∈Ω(g(n))

$0 \leq cn \leq (n-1)$

$0 \leq c \leq \frac{n^2-2n+1}{n^2}$

Se n0=2:

$0 \leq c \leq \frac{2^2-2*2+1}{2^2}$

$0 \leq c \leq \frac{2^2-2*2+1}{2^2}$

$0 \leq c \leq \frac{1}{4}$

Agora, fazendo a analise para ∞ :

$\lim_{x\to\infty} \frac{n^2-2n+1}{n^2} \quad = 1$

$\frac{1}{4} - 1 = -\frac{3}{4}$

$c = -\frac{3}{4}$

$0 \leq -\frac{3}{4} * n \leq \frac{n^2-2n+1}{n^2}, \forall n \geq 2$

Como a afirmativa anterior é verdadeira, f(n) = Ω(g(n)

1.3 $f(n) = \Theta(g(n))$

Sabendo que a Notação Θ(g(n)) = f(n) : existe constantes positivas C1, c2 e N0, tais que $0 \leq c1g(n) \leq f(n) \leq c2 * g(n), \forall n \geq n0$

f(n) = (x-1)$^{\mathbf{2}}$ ∈Θ (n$^{\mathbf{2}}$)

$0 \leq c1 * n \leq (x-1) \leq c2n$

$0 \leq c1 \leq \frac{n^2-2n+1}{n^2} \leq c2 * n$

$0 \leq c1 \leq \frac{n^2-2n+1}{n^4} \leq c2$

Para n0=2:

$\frac{2-2*2+1}{n^4} = \frac{1}{16}$

Analisando para infinito

$\lim_{x\to\infty} \frac{n^2-2n+1}{n^4} = 0$

$0 \leq \frac{1}{16} \leq n^2 - 2n + 1n^4 \leq 0$

Como a afirmação acima não existe, f(n) = Θ(g(n) é falsa.

2 Considere a seguinte função que recebe um valor n ≥ 1.

```
int loops(int n){
1 int i,j,r=0;
2 for(i=0; i<n; i++)
3 for(j=0; j<i; j++)
4 r+=i+j;
5 return r;
}
```

2.1 Determine a função f(n) que descreve a quantidade de execuções da linha 4. Expresse sua resposta em função de n utilizando somatório e resolva-o para obter f(n). Forneça a complexidade do tempo de execução do pior caso usando a notação O.

Para descrever a quantidade de execuções:

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

Obs:: $\sum_{j=0}^{i-1} 1 = \sum_{j=0}^{i-1} 1 = 1 + \sum_{j=1}^{i=1} = 1 + 1 + 2 + 3 + ... + i - 2 + i - 1 = i - 1 + 1 = i$

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

Logo, f(n) = $\frac{(n-1)n}{2}$

Sabendo que a Notação O(g(n)) = f(n) : existe constantes positivas C1 e N0, tais que $0 \leq f(n) \leq c * g(n), \forall n \geq n0$

Dado f(n) = $\frac{(n-1)n}{2}$:

Supondo : g(n) = n$^{\mathbf{2}}$, $\frac{(n-1)n}{2}$ ∈O(n^2) ?

$0 \leq \frac{(n-1)n}{2} \leq c * n$

$0 \leq \frac{n-1}{2} \leq c$

Adotando um N0=2, pois é necessário N0 > 0.

$0 \leq \frac{2-1}{2} \leq c$

$0 \leq \frac{1}{2} \leq c$
$0 \leq \frac{n-1}{2} \leq \frac{1}{2} * n, \forall n \geq 2$
Portanto f(n) = O(n^2)

2.2 Qual valor de r a função loops retorna? Expresse sua resposta em função de n utilizando somatório e resolva-o para obter r(n). Mostre a complexidade de r(n) usando a notação O.

A função para descrever o somatorio de r(n) é:
$\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} (i + j)$

Adotando para N=5:
Interação 0:
Não entra no segundor for, não realiza o calculo, então r=0
Interação 1:
r = 1 + 0 + 0 = 1
Interação 2:
r = 2 + 0 + 1 + 2 + 1 + 3 = 6
Interação 3:
r = 3 + 0 + 6 + 3 + 1 + 9 + 3 + 2 + 13 = 18
Interação 4:
r = 4 + 0 + 18 + 4 + 1 + 21 + 4 + 2 + 27 + 4 + 3 + 33 = 40
Resultando na fórmula:
$f(n) = \frac{(n-1)^2 * n}{2}$
Agora analizando a complexidade para O(n):
$\frac{(n-1)^2 * n}{2} = O(n)?$
Sabendo que a Notação O(g(n)) = f(n) : existe constantes positivas C1 e N0, tais que $0 \leq f(n) \leq c * g(n), \forall n \geq n0$
Tendo f(n) = $\frac{(n-1)^2 * n}{2}$ e g(n) = n^3:
$0 \leq \frac{(n-1)^2 * n}{2} \leq c * n$
$0 \leq \frac{(n^2-n+1) * n}{2} \leq c * n$
$0 \leq \frac{(n^3-n^2+n) * n}{2} \leq c * n$
$0 \leq \frac{\frac{n^3}{n^3} - \frac{n^2}{n^3} + \frac{n}{n^3}}{2} \leq c * \frac{n^3}{n^3}$
$0 \leq \frac{1 - \frac{1}{n} + \frac{1}{n^3}}{2} \leq c$
Sendo n0 = 1 , $c = \frac{1}{2}$
$0 \leq \frac{1}{2} \frac{n^2-2n+1}{n^2} \leq \frac{1}{2}, \forall n \geq 1$
Portanto f(n) é O(n^3)

3 Prove, usando indução, que 1+3+5+...+(2n-1) = n2, para n1.

Caso base:
Para n=1. 1^2 = 1, isso sendo verdade, podemos passar para o passo de indução.
Como n=1, vamos verificar se para n= k+1
1 + 3 + 5 + ... + (2n 1) + (2n + 1) = (n + 1)2, n $\leq$ 1
1 + 3 + 5 + . . . + (2n 1) + (2n + 1) = n^2 + (2n + 1) = $(n + 1)^2$

4 Considere o código abaixo que recebe dois números naturais.

```
F(n,m){
i=0; x=0;
while(i<n){
i = i+1;
x = x+m;
}
return x;
}
```

4.1 O que este algoritmo calcula? Qual invariante de laço este algoritmo mantém?

O algoritmo calcula o produto de n e m. A invariante de laço neste caso é que no final de cada iteração do laço, a variável x contém a soma dos elementos da interação 1 à interação n-1.

4.2 Mostre, por invariante de laço, que este algoritmo é correto.

Sabendo que a variante de laço:

"A invariante de laço de F(n,m) é que no final de cada iteração do laço, a variável x contém a soma dos elementos da interação 0 à interação n-1."

4.2.1 Inicio

Para F(1):

Antes da primeira interação, i=0 e como $1 < 0$, o algoritmo entra no laço e reproduz a soma na variavel x, provando que para o caso base é verdade.

4.2.2 Manutenção

Como no passo i = i +1, continua mantendo que entre no laço e executa o calculo de x = x+m, mantendo a invariante de laço verdadeira.

4.2.3 Término

Foi realizado todas as interações do laço, efetuando todas as somas, e resultando no valor de x, que é o produto de n e m.

Sendo assim, o laço se inicia realizando as somas, de x = x + m, até que chegue em n, e assim encerrando o laço retornando x, que é o produto.

5 Considere o código abaixo que recebe dois números naturais a e b.

```
A(a,b){
x=0;
while(b > 0){
if(b % 2 == 1) x = x + a;
a = 2 * a;
b =  b  / 2  ;
```

```
}
return x;
}
```

5.1 O que este algoritmo calcula? Qual invariante de laço este algoritmo mantém?

Este algoritmo tem como objetivo descobrir a quantidade de divisões que b faz por 2, 2*n*a e na pior hipotese caso b seja ímpar, irá ser $\sum_{i=1}^{n} = 2a - 1$. Já para a invariante de laço é a quantidade sucessiva de vezes que b é dividido por 2.

5.2 Mostre, por invariante de laço, que este algoritmo é correto.

Para b par, ele continuará o laço até 0 e dá o valor de x.

Para b ímpar, ele também irá continuará no laço até 0, também retornando o valor de x, logo o algoritmo está correto.

Por exemplo, com b=13 e a=1:

5.2.1 Interação 1:

13 > 0
Verificando 13 % 2 = 1, sendo verdade,
x = 0 + 1 = 1
a = 2 * 1 = 2
$b = \frac{13}{2} = 6$

5.2.2 Interação 2:

6 > 0
Verificando 5 %2 =0, sendo falso, não se altera x
a = 2 * 2 = 4
$b = \frac{6}{2} = 3$

5.2.3 Interação 3:

3 > 0
Verificando 3 %2=1, sendo verdadeiro,
x = 1 + 4 = 5
a = 2 * 2 = 4

$b = \frac{3}{2} = 1$

5.2.4 Interação 4:

1 > 0

Verificando 1%2 =1 , sendo verdade,
x = 5 + 4 = 9 a = 2 * 4 = 8
$b = \frac{1}{2} = 0$
Como b=0, o algortimo se conclui e retorna 9.