Nome: Victor Akio Domoto RA: 150546 Turma: I

Projeto e Análise de Algoritmos – Lista de exercícios 1

1) A)

Projeto e Análise de Algoritmos
Lista de Exercícios 1

(1) $f(n) = (n+1)^2$ e $g(n) = n^2$

(a) $f(n) = O(g(n)) \Rightarrow O(g(n)) = \{f(n) \mid \exists^+ c, n_0 \mid 0 \leq f(n) \leq c \cdot g(n), \forall n \geq n_0\}$.

$f(n) = O(g(n)) \Rightarrow 0 \leq f(n) \leq c \cdot g(n)$

$\Rightarrow 0 \leq (n+1)^2 \leq c \cdot n^2 \Rightarrow 0 \leq n^2 + 2n + 1 \leq c \cdot n^2 \Rightarrow \dfrac{0 \leq n^2 + 2n + 1 \leq c \cdot n^2}{n^2}$

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

— Resolvendo a inequação da esquerda:

$0 \leq 1 + \frac{2}{n} + \frac{1}{n^2}$ * Note que o único jeito dessa inequação ser falsa é para $n < 0$. Entretanto, pela própria definição de $O(g(n))$, temos que n é uma constante positiva. Logo, a inequação é verdadeira $\forall n \geq n_0$.

— Resolvendo o lado direito da inequação para encontrar c e n_0:

* $n = 1 \Rightarrow 1 + \frac{2}{1} + \frac{1}{1^2} \leq c \Rightarrow 1 + 2 + 1 \leq c \Rightarrow 4 \leq c \Rightarrow \boxed{c = 4 \text{ para } n \geq 1}$

* Portanto, temos que $0 \leq f(n) \leq c \cdot g(n)$ é verdadeiro para $c = 4$ e $n_0 = 1$.

$\Rightarrow 0 \leq (n+1)^2 \leq 4n^2, \forall n \geq n_0 = 1 \Rightarrow f(n) = O(g(n))$. ∎

1) B)

(b) $f(n) = \Omega(g(n)) \Rightarrow 0 \leq c \cdot g(n) \leq f(n)$

— Testando a inequação para $c = 1$:

$0 \leq 1 \cdot n^2 \leq (n+1)^2 \Rightarrow 0 \leq n^2 \leq n^2 + 2n + 1$

* Pela definição de $\Omega(g(n))$, temos $n \geq n_0$ onde n_0 é uma constante positiva. Logo, $\forall n \geq 1$, $0 \leq n^2 \leq n^2 + 2n + 1$.

$\therefore c = 1, n_0 = 1, 0 \leq n^2 \leq (n+1)^2 \; \forall n \geq n_0 \Rightarrow f(n) = \Omega(g(n))$. ∎

1) C)

(c) $f(n) = \Theta(g(n)) \Rightarrow 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n), \ \forall\, n \geq n_0.$

- Para provar isso, é necessário separar essa inequação em 2:

(1) Inequação da esquerda:

$0 \leq c_1 g(n) \leq f(n) \Rightarrow 0 \leq c_1 \cdot n^2 \leq n^2 + 2n + 1$

* Testando para $c_1 = 1$:

$0 \leq 1 \cdot n^2 \leq n^2 + 2n + 1 \Rightarrow$ essa inequação é verdadeira, uma vez que $n > 0$ pela própria definição de Θ. Portanto, para $\boxed{c_1 = 1}$ e $n \geq 1$, $0 \leq c_1 \cdot n^2 \leq (n+1)^2 \ \forall\, n \geq, n_0 = 1$.

(2) Inequação da direita:

$f(n) \leq c_2 \cdot g(n) \Rightarrow n^2 + 2n + 1 \leq c_2 \cdot n^2 \Rightarrow \dfrac{(n^2 + 2n + 1)}{n^2} \leq \dfrac{(c_2 \cdot n^2)}{n^2}$

$\Rightarrow \boxed{1 + \dfrac{2}{n} + \dfrac{1}{n^2} \leq c_2}$

* Testando a inequação (2):

$n = 1 \Rightarrow 1 + \dfrac{2}{1} + \dfrac{1}{1^2} \leq c_2 \Rightarrow 4 \leq c_2 \Rightarrow \boxed{c_2 = 4 \text{ e } n \geq, n_0 = 1}$

$\therefore$ para $c_1 = 1, c_2 = 4$ e $n_0 = 1$, $0 \leq c_1 \cdot n^2 \leq (n+1)^2 \leq c_2 \cdot n^2$

$\Rightarrow 0 \leq n^2 \leq (n+1)^2 \leq 4n^2 \ \forall\, n \geq 1 \Rightarrow f(n) = \Theta(g(n))$. ■

2) A)

(2) (a) for (i=0; i<n; i++)
for(j=0; j<i; j++)
r += i + j;

- Seja $f(n)$ uma função que determina a quantidade de execução da linha 4:

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

$\Rightarrow f(n) = \sum_{i=0}^{n-1} i = \boxed{\dfrac{n \cdot (n-1)}{2}}$ //

- $f(n) = O(n^2) \Rightarrow 0 \leq f(n) \leq c \cdot n^2$

* Provando que $f(n) \in O(n^2)$

$0 \leq \dfrac{n^2 - n}{2} \leq c \cdot n^2 \Rightarrow 0 \leq \left(\dfrac{\frac{(n^2-n)}{2}}{n^2} \right) \leq \dfrac{(c \cdot n^2)}{n^2}$

$\Rightarrow \boxed{0 \leq \dfrac{1}{2} - \dfrac{1}{2n} \leq c}$ * Note que para $n = 1$, $f(n)$ é zero. Portanto, $n \geq 2 \Rightarrow n_0 = 2$.

* Verificando $f(n)$ para n suficientemente grande:

$n \to \infty = \lim_{n \to \infty} \dfrac{1}{2} - \cancelto{0}{\dfrac{1}{2n}} = \dfrac{1}{2} \Rightarrow \boxed{c = \dfrac{1}{2}}$

$\Rightarrow$ para $c = \frac{1}{2}$ e $n_0 = 2$, $0 \leq c \cdot n^2 \leq \dfrac{n \cdot (n-1)}{2} \ \forall\, n \geq 2$.

$\Rightarrow f(n) = O(n^2)$. ■

2) B)

② (b) $r(n) = \sum_{i=0}^{n-1}\sum_{j=0}^{i-1}(n-1) \Rightarrow \sum_{i=0}^{n-1}(i(n-1)) \Rightarrow r(n) = \sum_{i=0}^{n-1}(i \cdot n) - \sum_{i=0}^{n-1} 1$

$\Rightarrow r(n) = \frac{n^2(n-1)}{2} - \sum_{i=0}^{n-1} 1 \Rightarrow r(n) = \left(\frac{n^2(n-1)}{2}\right) - \left(\frac{n(n-1)}{2}\right) \Rightarrow r(n) = \left(\frac{n^3-n^2}{2}\right) - \left(\frac{n^2-n}{2}\right)$

$\Rightarrow \boxed{r(n) = \frac{n^3-2n^2+n}{2}} \Rightarrow r(n) = O(n^3)$

* Provando que $r(n) = O(n^3)$

$r(n) = O(n^3) \Rightarrow 0 \leq r(n) \leq c \cdot n^3$

$0 \leq \frac{n^3-2n^2+n}{2} \leq c \cdot n^3 \Rightarrow 0 \leq \left(\frac{\left(\frac{n^3-2n^2+n}{2}\right)}{n^3}\right) \leq \frac{(c \cdot n^3)}{n^3}$

$\Rightarrow \boxed{0 \leq \frac{1}{2} - \frac{1}{n} + \frac{1}{2n^2} \leq c}$ * Note que para $n = 2$, $r(n)$ será zero. Assim, $n \geq 2 \Rightarrow \boxed{n_0 = 2}$

* Verificando para n suficientemente grande:

$n \to \infty = \lim_{n \to \infty} \frac{1}{2} - \overset{0}{\cancel{\frac{1}{n}}} + \overset{0}{\cancel{\frac{1}{2n^2}}} = \frac{1}{2} \Rightarrow \boxed{c = \frac{1}{2}}$

$\therefore$ para $c = \frac{1}{2}$ e $n_0 = 2$, $0 \leq \left(\frac{n^3-2n^2+n}{2}\right) \leq c \cdot n^3 \ \forall n \geq 2. \Rightarrow r(n) = O(n^3)$. ■

3)

(3) $1+3+5+\dots+(2n-1) = n^2 \ \forall n \geq 1 \in \mathbb{N}$

* Base: provar que para $n=1$, a proposição é verdadeira:

$n=1 \Rightarrow 2n-1 = n^2 \Rightarrow 2\cdot(1)-1 = (1)^2 \Rightarrow \boxed{1=1}$

* Hipótese indutiva: assumindo que se $n=k$ é verdadeiro, tal que $k \in \mathbb{N}$ e $k \geq 1$, $n=k+1$ também deve ser:

$1+3+5+\dots+(2k-1) = k^2$

* Pero: $1+3+5+\dots+(2k-1)+(2(k+1)-1)$, de modo que $k^2 = 1+3+5+\dots+(2k-1)$

$\Rightarrow k^2+(2\cdot(k+1)-1) = k^2+(2k+2-1) \Rightarrow \boxed{k^2+2k+1}$

* Note, ainda, que temos um quadrado perfeito $(a+b)^2 = a^2+2\cdot ab+b^2$

$\Rightarrow k^2+2k+1 = (k+1)^2$

$\therefore$ pela hipótese da indução de que se $n=k$ for verdadeiro então $n=k+1$ também deve ser, temos que essa proposição é verdadeira $\forall n \in \mathbb{N}$ e $n \geq 1$. ■

4) A) B)

(4) (a) Este algoritmo, que possui um laço do tipo while, repete a operação $x' = x + m$ n vezes, onde x' é o valor atualizado de x e i uma variável contadora (a cada iteração). Sendo assim, temos que o algoritmo calcula a multiplicação entre dois números (n e m) através da repetição (while).

- Invariante de laço: $x = m \cdot i$, "a variável X respeitará essa condição antes, durante e após a execução do laço.

(b) Demonstração: $x = m.i$

- Inicialização: note que antes da primeira iteração, temos que $X = 0$, $i = 0$ e m é uma entrada qualquer. Sendo assim, temos: $x = m.0 \Rightarrow \boxed{x=0}$. Portanto, o invariante é satisfeito.

- Manutenção:

  * Dentro do laço while há exatamente 2 operações, $i' = i+1$ e $x' = x + m$, de modo que i' e x' são os valores atualizados de x e i a cada iteração.

  * Primeira iteração: note que a primeira operação é $i' = i+1$, ou seja, $i = 1$. Isso implica que se $m \neq 0$, então $x_1 = 0 + m$, que é igual o invariante escolhido, $x = m \cdot i$, onde $i = 1 \Rightarrow x = m$.

  * Segunda iteração: novamente a primeira operação é $i' = i+1$, mas dessa vez temos $i = 2$. Como na 1º iteração $x = m$, temos que $x_2 = x + m \Rightarrow x_2 = m + m \Rightarrow x_2 = 2.m$. Observe que esse resultado satisfaz o invariante $x = m.i$, onde $i = 2 \Rightarrow x = m.2 \Rightarrow x = 2m$.

  * n-1 iterações: as outras iterações irão seguir exatamente a mesma ideia, como uma indução matemática.

- Terminação:

  * O laço só termina quando $i \geq n$. Entretanto, como i só varia 1 unidade por iteração, temos que o laço acaba quando $i = n$.

  Portanto, pelo invariante de laço $x = m \cdot i$, temos que $i = n$. Logo, $x = m.n$, que é exatamente a multiplicação entre dois números (n e m), objetivo final do algoritmo.

  Dessa forma, temos que esse algoritmo é correto. ■

5) A) B)

(5)(a) Este algoritmo basicamente calcula a multiplicação entre dois números através de 3 operações dentro de um laço.

– Invariante do laço: $ab = a'b' + X$, de modo que $a'b'$ é a atualização dos valores de a e b a cada iteração do laço. "O produto ab respeitará essa condição antes, durante e após a execução do laço.

(b) Demonstração: $ab = a'b' + X$

– Inicialização:

* Antes de entrar no laço while, temos a atribuição $x=0$. Se $x=0$, então $ab = a'b' + 0$ $\Rightarrow ab = a'b'$, onde $a'b'$ ainda não sofreu alterações, logo o invariante é satisfeito.

– Manutenção:

* Durante a execução do laço, por ter uma estrutura de decisão, significa que temos duas situações distintas a cada iteração.

* If $(b\%2 == 1)$ then: $x = x + a$;. Antes de olhar para essa estrutura, observe que durante a execução do laço, temos duas operações: $a = 2a$ e $b = \lfloor b/2 \rfloor$. Entretanto, como se trata de divisão de números inteiros, esse arredondamento para o piso acaba perdendo informação, que é exatamente quando o resto da divisão é igual a 1, como por exemplo, $b = 5 \Rightarrow b = 5/2 = 2$ e sobra 1. Portanto, a estrutura de decisão serve exatamente para corrigir esse erro, pois quando perdemos um resto $= 1$, o laço perde uma iteração, que é exatamente a atualização do valor a, por isso $X' = X + a$.

* Pelo invariante do laço: $ab = a'b' + X$

* 1ª iteração: $a = 2a$ e $b = \lfloor \frac{b}{2} \rfloor \Rightarrow ab = (2 \cdot a) \cdot \lfloor \frac{b}{2} \rfloor + X$, note que toda vez que $(b\%2 == 1)$, o $2a$ está sendo multiplicado por -1 vez do que deveria. Então com a soma de X, o $\lfloor \frac{b}{2} \rfloor$ deixa de ser um arredondamento e passa a ser exato $\Rightarrow ab = (2 \cdot a) \cdot \left(\frac{b}{2}\right) \Rightarrow \boxed{ab = ab}$

* 2ª iteração: segue a mesma ideia da 1ª: $ab = 4 \cdot a \cdot \lfloor \frac{b}{4} \rfloor + X$, quando somamos X $\lfloor \frac{b}{4} \rfloor$ deixa de ser um arredondamento. Então $ab = 4 \cdot a \cdot \frac{b}{4} \Rightarrow \boxed{ab = a \cdot b}$.

– Terminação:

* O laço só terminará quando $b \leq 0$. Entretanto, note que como $b = \lfloor \frac{b}{2} \rfloor$, então para qualquer b, ele sempre acabará em $b = 0$.

* Assim, temos pelo invariante do laço: $ab = a'b' + X$, onde $b = 0 \Rightarrow ab = a' \cdot (0) + x \Rightarrow \boxed{ab = x}$, onde X retornará o valor da multiplicação entre a e b. Logo, o algoritmo está correto. ∎