DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Ejercicios 3.1 y 3.2

Algoritmos y Estructuras de Datos II

Integrante	LU	Correo electrónico
Castro Russo, Matias Nahuel	203/19	castronahuel14@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

a) Sea b ∈ R tal que b ≥ 2. Demostrar que n ≤ b^n para todo n ∈ N, por inducción en n.

Empiezo mirando el Caso Base, tomo $p(n) = n \leq b^n$:

n = 1) n ≤ $b^n \Leftrightarrow 1 \leq b^1 \Leftrightarrow 1 \leq b$

Esto se cumple siempre ya que b ≥ 2. Por lo tanto $1 < 2 \leq b$

Entonces el caso base vale, ahora hago el Paso Inductivo:

Mi Hipótesis Inductiva es que vale $p(n)$ y quiero ver si $p(n) \Rightarrow p(n+1)$

Quiero ver que:

$$n + 1 \leq b^{n+1}$$

$$\Leftrightarrow n + 1 \leq b^n b$$

$$\Leftrightarrow n \leq b^n b - 1$$

$$\Leftrightarrow n \leq b^n b - 1$$

Se que n ≤ b^n por H.I

Se que $2 b^n - 1 \leq b b^n - 1$ ya que $2 \leq b$

Se que $b^n \leq 2 b^n - 1$ ya que $1 < b^n$ entonces $b^n + 1 \leq b^n + b^n$

Por lo tanto, uniendo toda esa información llego a que:

$$\Leftrightarrow n \leq b^n \leq 2 b^n - 1 \leq b b^n - 1$$

entonces por H.I, es correcto decir que $n \leq b^n b - 1 \Leftrightarrow n + 1 \leq b^{n+1}$

Por lo tanto, por inducción, afirmo que $p(n) \Rightarrow p(n+1)$

b) Sea como en el ítem anterior b ∈ R tal que b ≥ 2. Demostrar que $x \leq b^{x+1}$ para todo x ∈ R ≥ 0. Notar que ahora x ∈ R ≥ 0 es un número real no negativo.

Quiero ver que: $x \leq b^{x+1}$

Se que $x \leq \lfloor x \rfloor + 1$

Se que $\lfloor x \rfloor \leq x$

Entonces $b^{\lfloor x \rfloor + 1} \leq b^{x+1}$, pues $x + 1 \leq x + 1$ y $x + 1 \geq 1$

Uniendo la informacion que tengo y acotando, llego a:

$$x \leq \lfloor x \rfloor + 1 \leq b^{\lfloor x \rfloor + 1} \leq b^{x+1}$$

Donde $(\lfloor x \rfloor + 1) \in$ N .

Por lo tanto $\lfloor x \rfloor + 1 \leq b^{\lfloor x \rfloor + 1}$ vale por lo demostrado en a), siendo $\lfloor x \rfloor + 1$ lo que en el ítem anterior llamaba n, siendo n ∈ N .

Al ser $x \leq \lfloor x \rfloor + 1$ y $b^{\lfloor x \rfloor + 1} \leq b^{x+1}$

Entonces por Teorema del Sandwich, queda demostrado que $x \leq b^{x+1}$.

c) Sea ahora b ∈ R tal que b > 1. Como b > 1, sabemos que existe una constante k ∈ N tal que $b^k \geq 2$. Demostrar que $x / k \leq b^{x+k}$ para todo x ∈ R ≥ 0

Parto de lo demostrado en el ítem b), y voy a querer llegar a c)

Entonces tengo $x \leq b^{x+1}$ (pero en b), $b \geq 2$) pero ahora en ítem c) $b > 1$

Hago un renombre, b del ítem b), ahora la denoto como b^k

$$x \leq b^{k^{x+1}} \Leftrightarrow x \leq b^{k\,(x+1)}$$

$$\Leftrightarrow x \leq b^{kx+k} \Leftrightarrow x \leq b^k\, b^{kx}$$

Se que $x \in R$, $k \in N$.
Eso implica que su producto ∈ R, entonces $kx \in R \geq 0$

Hago otro renombre $x' = kx$
Donde $x \in R$, $k \in N \Rightarrow x' \in R \geq 0$

$$\Leftrightarrow x \leq b^k\, b^{kx} \Leftrightarrow x \leq b^k\, b^{x'}$$

Multiplicó por k de ambos lados y renombro:

$$\Leftrightarrow kx \leq b^k\, b^{x'}\, k \Leftrightarrow x' \leq b^k\, b^{x'}\, k$$

Ahora divido por k, notar que al multiplicar y dividir k, se que $k > 0$

$$\Leftrightarrow x'/k \leq b^k\, b^{x'}$$

Partí de b) y llegue a c). Por lo tanto queda demostrado que $x \leq b^{x+1} \Rightarrow x'/k \leq b^k\, b^{x'}$

Donde x' vendría a ser lo “mismo” que x, ambos $\in R \geq 0$

d) Sean como en el ítem anterior b ∈ R tal que b >1 y k ∈ N tales que b k ≥ 2. Demostrar que para todo x ∈ R ≥ 0 y para todo n,p ∈ N vale la siguiente desigualdad: $(x / pk)^n \leq b^{n\,(x/p+k)}$

Lo hago por Inducción, empiezo planteando el caso base:

Caso Base) con $n = 1$, $\quad (x / \ pk)^1 \leq b^{1\ (x/p+k)} \ \Leftrightarrow \ (x / \ pk) \ \leq b^{\ (x/p+k)}$

Donde $x \in R \geq 0 \ \wedge \ x \in N \ \Rightarrow \ x/p \in R \geq 0$

Llamo $x' = x / p$

$$\Leftrightarrow \ (x'/ k) \ \leq b^{\ (x'+k)}$$

Eso vale por c), entonces se que el caso base se cumple

P.I)
Hago el paso inductivo, mi H.I es que vale $p(n) \ = (x / \ pk)^n \leq b^{n\ (x/p+k)}$

Quiero ver si $p(n) \ \Rightarrow \ p(n+1)$

$$\Leftrightarrow (x / \ pk)^{n+1} \leq (b^{\ (x/p+k)})^{n+1}$$

$$\Leftrightarrow (x / \ pk)^n \ (x / \ pk) \ \leq (b^{\ (x/p+k)})^n \ (b^{\ (x/p+k)})$$

Se que todo es positivo, pues es todos los términos son Naturales o Reales mayores o iguales a cero
Entonces multiplicar y dividir de ambos lados con tranquilidad

$$\Leftrightarrow \ (x / \ pk) / \ (b^{\ (x/p+k)}) \ \leq (b^{\ (x/p+k)})^n \ / (x / \ pk)^n$$

Por mi Hipótesis inductiva, se que $(x / \ pk)^n \leq b^{n\ (x/p+k)}$

Lo cual implica que $(b^{\ (x/p+k)})^n \ / (x / \ pk)^n \ \geq 1$, pues el dividendo es mayor o igual al divisor

Por otro lado, se que $(x'/ k) \ \leq b^{\ (x'+k)}$ Por ítem c).

Por lo tanto, $(x / \ pk) / \ (b^{\ (x/p+k)}) \ \leq 1$, pues el dividendo es menor o igual al divisor

Entonces la siguiente desigualdad es verdadera:

$$\begin{array}{cc} (x / \ pk) / \ (b^{\ (x/p+k)}) & \leq (b^{\ (x/p+k)})^n \ / (x / \ pk)^n \\ \uparrow \leq \ 1 & \uparrow \geq \ 1 \end{array}$$

Es cierto que algo mayor o igual a 1, es mayor o igual, a algo que sea menor o igual a 1

e)

Sea g : R→R.
Entonces:O(g) def = { f : R→R | (∃ x_0 ∈ R , c ∈ R > 0) f(x) ≤ c * g (x) ∀ x ≥ x_0 }

llamo g(x) = b^x

llamo f (x) = x^p

quiero ver que f(x) ≤ c . g (x)

$$\Leftrightarrow x^p \leq c \,.\, b^x$$

Por otro lado, de d) saco la siguiente información

d) = $(x / pk)^n \leq b^{n\,(x/p+k)}$

uso p = n

$$(x / pk)^p \leq b^{p\,(x/p+k)}$$

$$\Leftrightarrow (x / pk)^p \leq b^{x+pk}$$

$$\Leftrightarrow (x / pk)^p \leq b^x\, b^{pk}$$

$$\Leftrightarrow (x)^p / (pk)^p \leq b^x\, b^{pk}$$

$$\Leftrightarrow (x)^p \leq b^x \,.\, b^{pk}\,(pk)^p$$

$$\Leftrightarrow (x)^p \leq b^{pk}\,(pk)^p \,.\, b^x$$

Eso es $f(x) \leq c \,.\, g(x)$

Donde el c existe, $c = b^{pk}\,(pk)^p$

y existe un x_0 , por ejemplo con $x_0 = 0$ vale

Por lo tanto, $x^p \in O(b^x)$

```
FUNCTION P(A : arreglo(nat)) ⟶ B : arreglo(nat)
1:  n:=tam(a)                                                          ▷ O(1) ∧ Ω(1)
2:  M:=0                                                               ▷ O(1) ∧ Ω(1)
3:  for i:=0 to n-1 do                                                 ▷ O(n) ∧ Ω(n)
4:      if  (A[I] >= N)  then                                          ▷ todo el if es O(1) ∧ Ω(1)
5:          A[i] := 0                                                  ▷ O(1) ∧ Ω(1)
6:      else
7:          M := max(M, A[i])                                          ▷ O(1) ∧ Ω(1)
8:  B := nuevo arreglo(nat) indexado desde 0 hasta M inclusive, inicializado en 0   ▷ O(n) ∧ Ω(1)
9:  for i:=0 to M do                                                   ▷ O(n) ∧ Ω(1)
10:     for j:=i to M do                                               ▷ O(n) ∧ Ω(1)
11:         B[A[i]] := 1 + B[A[i]] + B[A[j]]                           ▷ O(1) ∧ Ω(1)
12: return B                                                           ▷ O(n) ∧ Ω(1)
```

Conclusiones y aclaraciones

Complejidad Peor Caso : $\mathcal{O}$(n*n) = $\mathcal{O}(n^2)$

Complejidad Mejor Caso : Ω(n)

Mejor caso: El mejor caso seria donde en el primer for (el cual siempre se va a ejecutar n veces si o si,y todo lo que esta dentro del for es $\mathcal{O}(1)$) todos los elementos del arreglo sean mayores o iguales a n, o que sean 0. La cosa es que tendrian que entrar al primer if (A[i] >= n), o entrar al segundo, pero que sea cero. Con el fin de que al terminar el for, M sea igual a 0.

De esa manera, B se inicializaria con longitud = 0, por lo cual nunca entraria al 2do for, lo que valga la rebundancia, implicaria que tampoco entre al tercer for, ya que esta contenido en el segundo. De esa manera todo el resto de instrucciones son $\mathcal{O}(1)$, con el return inclusive, ya que devolver un arreglo vacio cuesta $\mathcal{O}(1)$ Por lo tanto, el mejor caso es $\mathcal{O}$(n)

Peor Caso: El peor caso es aquel donde el arreglo de entrada A, contiene todos valores menores estrictos a n , y que por lo menos uno sea n-1. Por lo cual, M = n-1, y B se inicializa con n elementos, lo cual pertenece a $\mathcal{O}$(n). En ese caso, el segundo for perteneceria a $\mathcal{O}$(n) por la complejidad de las operaciones internas, y el tercer for al ser $\mathcal{O}$(n), implica que el segundo for completo es $\mathcal{O}(n^2)$, devolver B, me cuesta O(n), y el resto de operaciones son $\mathcal{O}(1)$. Por lo tanto, el peor caso pertenece a $\mathcal{O}(n^2)$

EJERCICIO 3.2)

a) Escribir en castellano el invariante de representación.

1) e.seleccionada no pertenece a inactivasVacias, ni a inactivasNOVacias

2) toda pestaña de inactivasNOVacias, debe no pertenecer a inactivasVacias, y su string debe ser NO vacío

3) Todas las pestañas deben ser válidas, es decir que el conjunto de todas arranque en 0, y estén todas las pestañas consecutivamente como corresponde hasta llegar a n (n = número de pestañas totales del editor)

b) Escribir formalmente el invariante de representación.

(1) e.seleccionada $\in$ e.inactivasVacias $\wedge \neg$ ($\exists\ p \in$ e.inactivasNoVacias)($\pi_1(p)$ = e.seleccionada)

(2) $\neg$ ($\exists\ p \in$ e.inactivasNoVacias)($\pi_1(p) \in$ e.inactivasVacias $\vee$ vacía?($\pi_2(p)$))

(3) ($\forall\ i : nat$) ($0 \leq i <$ #e.inactivasVacias + #e.inactivasNoVacias + 1 $\Rightarrow_L i$ = seleccionada
$\vee\ i \in$ e.inactivasVacias
$\vee$ ($\exists\ p \in$ inactivasNOVacias) ($\pi_1(p) = i$))

$$rep(e) = (1) \wedge (2) \wedge (3)$$

c) Escribir formalmente la función de abstracción.

$abs(e) = p : Editor \mid$
#e.inactivasVacias + #e.inactivasNOVacias + 1 $=_{obs}$ #pestañas (p)
$\wedge_L$ ($\forall\ n : nat$) ($0 \leq i <$ #pestañas (p) $\wedge_L$ seleccionada $(p, n) \Rightarrow_L n =_{obs}$ e.seleccionada)
$\wedge_L$ ($\forall\ n : nat$) ($0 \leq i <$ #pestañas (p) $\wedge_L$ seleccionada $(p, n) \Rightarrow_L$
e.anteriores & e.posteriores $=_{obs}$ texto (p, n))
$\wedge$ long(e.anteriores) = posiciónCursor (p)
$\wedge$ ($\forall\ n : nat$) ($0 \leq n <$ #pestañas (p) $\wedge_L \neg$ seleccionada (p, n) $\wedge$ vacía?(texto (p, n)) $\Rightarrow_L n \in$
e .inactivasVacias)
$\wedge$ ($\forall\ n : nat$) ($0 \leq n <$ #pestañas (p) $\wedge_L \neg$ seleccionada (p, n) $\wedge \neg$ vacía?(texto (p, n)) $\Rightarrow_L$
$n \in$ e.inactivasNoVacias)
$\wedge_L$ ($\forall\ pest \in$ inactivasNoVacias) (texto $(pest, n) =_{obs} \pi_2(pest)$)

Como abuso de notación, omití poner los “sombreritos” de abstracción y en lógica de primer order $0 \leq i <$ #pestañas (p) tendría que hacerlo en dos partes y conectarlos logicamente, asi $0 \leq i \wedge i <$ #pestañas (p)