Guia 2

Ejercicios obligatorios de la práctica

31 de mayo de 2020 — Algoritmos y Estructuras de Datos II

Integrante	LU	Correo electrónico
Rodriguez, Miguel	57/19	mmiguerodriguez@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
https://exactas.uba.ar

Ejercicio 1: Funciones polinomiales vs exponenciales

(a) Sea $b \in \mathbb{R}$ tal que $b \geq 2$. Demostrar que $n \leq b^n$ para todo $n \in \mathbb{N}$, por inducción en n.

$$P(n) \equiv n \leq 2^n \ \forall n \in \mathbb{N} \tag{1}$$
$$\text{c.b. } P(1) \equiv 1 \leq 2^1 \equiv True \tag{2}$$
$$\text{sup. } P(n) \equiv n \leq b^n \tag{3}$$
$$\text{q.v.q. } P(n+1) \equiv n+1 \leq b^{n+1} \tag{4}$$
$$P(n+1) \equiv n+1 \leq b^n \times b \tag{5}$$
$$n \leq b^n \ \Rightarrow \ n \times b \leq b^n \times b \tag{6}$$
$$n+1 \leq n \times b \ \forall n, b \geq 2 \tag{7}$$
$$n+1 \leq b^{n+1} \tag{8}$$
$$P(n) \ \Rightarrow \ P(n+1) \quad \blacksquare \tag{9}$$

(b) Sea $b \in \mathbb{R}$ tal que $b \geq 2$. Demostrar que $x \leq b^{x+1}$ para todo $x \in \mathbb{R}_{\geq 0}$.

$$x \leq \lfloor x \rfloor + 1 \tag{10}$$
$$\text{Usamos (a) con } n = \lfloor x \rfloor + 1 \in \mathbb{N} \tag{11}$$
$$\lfloor x \rfloor + 1 \leq b^{\lfloor x \rfloor + 1} \tag{12}$$
$$\text{Ademas } x \leq \lfloor x \rfloor + 1 \leq x + 1 \tag{13}$$
$$x \leq \lfloor x \rfloor + 1 \leq b^{\lfloor x \rfloor + 1} \leq b^{x+1} \tag{14}$$
$$\Rightarrow \ x \leq b^{x+1} \quad \blacksquare \tag{15}$$

(c) Sean $b \in \mathbb{R}$ tal que $b > 1$ y $k \in \mathbb{N}$ tal que $b^k \geq 2$. Demostrar que $\frac{x}{k} \leq b^{x+k}$ para todo $x \in \mathbb{R}_{\geq 0}$.

$$\frac{x}{k} \leq b^{k(\frac{x}{k}+1)} \tag{16}$$
$$\text{Sabemos que } b^k \geq 2 \text{, renombramos } b^k = b \tag{17}$$
$$\text{Hacemos lo mismo con } \frac{x}{k} \text{ ya que } x \in \mathbb{R}_{\geq 0},\ k \in \mathbb{N} \text{, renombramos } \tfrac{x}{k} = x \tag{18}$$
$$\text{Se cumple que } b \in \mathbb{R} \geq 2 \text{ y } x \in \mathbb{R}_{\geq 0} \text{ (hipótesis de (b))} \tag{19}$$
$$x \leq b^{x+1} \text{ (vale por (b))} \quad \blacksquare \tag{20}$$

(d) Sean $b \in \mathbb{R}$ tal que $b > 1$ y $k \in \mathbb{N}$ tal que $b^k \geq 2$. Demostrar que $\forall x \in \mathbb{R}_{\geq 0}$ y $\forall n, p \in \mathbb{N}$ vale la siguiente desigualdad: $(\frac{x}{pk})^n \leq b^{n(\frac{x}{p}+k)}$.

$$P(n) \equiv \left(\frac{x}{pk}\right)^n \leq b^{n(\frac{x}{p}+k)} \tag{21}$$
$$\text{c.b. Tomamos } \tfrac{x}{p} = j, j \in \mathbb{R}_{\geq 0} \tag{22}$$
$$P(1) \equiv \frac{j}{k} \leq b^{j+k} \text{ (vale por (c))} \tag{23}$$
$$\text{sup. } P(n) \tag{24}$$
$$\text{q.v.q. } P(n+1) \equiv \left(\tfrac{x}{pk}\right)^{n+1} \leq b^{(n+1)(\frac{x}{p}+k)} \tag{25}$$

Veamos que evaluar P(n + 1) es lo mismo que multiplicar ambos lados por sus mismos factores, entonces si la desigualdad se cumplia, al multiplicar ambos lados por lo mismo se va a seguir cumpliendo. $\blacksquare$

(e) Demostrar que, vistas como funciones de $x \in \mathbb{R}_{\geq 0}$, vale: $x^p \in O(b^x)$ para toda base $b \in \mathbb{R}$ tal que $b > 1$ y para todo exponente $p \in \mathbb{N}$.

$$f(x) = x^p \wedge g(x) = b^x \tag{26}$$
$$\text{q.v.q. } (\exists x_0 \in \mathbb{N},\ c \in \mathbb{R}_{\geq 0})\ f(x) \leq c.g(x)\ \forall x \geq x_0 \tag{27}$$
$$\text{Usamos (d) y tomamos } n = p \tag{28}$$
$$\left(\frac{x}{pk}\right)^p \leq b^{p(\frac{x}{p}+k)} \tag{29}$$
$$\text{Tomamos } \tfrac{x}{p} = x, x \in \mathbb{R}_{\geq 0} \tag{30}$$
$$\frac{x^p}{k^p} \leq b^{px} \times b^{pk} \tag{31}$$
$$x^p \leq b^{px} \times b^{pk} \times k^p \tag{32}$$
$$\text{Veamos que aca podemos tomar } k^p \text{ y } b^{pk} \text{ como constantes} \tag{33}$$
$$x^p \leq b^{p^x} \times c_1 \times c_2 \tag{34}$$
$$\text{Tomamos } b^p = b \tag{35}$$
$$x^p \leq b^x \times c \tag{36}$$
$$\text{Tomando } x_0 = x \text{ vale que } x^p \in O(b^x) \quad \blacksquare \tag{37}$$

Ejercicio 2: Complejidades

```
function P(A : arreglo(nat)) -> B : arreglo(nat) {
  n := tam(A)
  M := 0
  for i := 0 to n - 1 {
    if (A[i] >= n) { A[i] := 0 } else { M := max(M, A[i]) }
  }
  B := nuevo arreglo(nat) desde 0 hasta M inclusive, inicializado en 0
  for i := 0 to M {
    for j := i to M {
      B[A[i]] := 1 + B[A[i]] + B[A[j]]
    }
  }
  return B
}
```

En esta funcion, tenemos 2 variables importantes

1. n: Tamaño del arreglo A
2. M: Elemento mas grande en el arreglo A, menor a long(A). Está acotado. ($0 \leq$ M $\leq$ n - 1)

Sabemos que la primer iteracion va a ser Ω(n) $\wedge$ O(n) = Θ(n) ya que no importa que valores tenga n, siempre se va a iterar todo el tamaño del arreglo A.

Generar el nuevo arreglo B inicializado en 0 es Θ(M) pero va a estar acotado por la segunda iteracion.

Ademas, la segunda iteracion depende del M.

1. Mejor caso: Ocurre cuando el segundo ciclo no se ejecuta nunca, es decir que el elemento más grande en el arreglo A sea 0, o que A sea una lista con todos ceros. A = [0, 0, 0 ... , 0].

$$\sum_{i=0}^{M=0} 1 = 0 = \Omega(0)$$

2. Peor caso: el elemento más grande del arreglo A es igual a n - 1 entonces vamos a tener M = n - 1. Reemplazando el M por su peor caso para calcular la complejidad nos da una sumatoria ya que en el primer for iteramos de 0 a M, y en el segundo desde i hasta M.

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

Sumamos los ordenes de mejor y peor caso de cada iteracion

$\Rightarrow$ P $\in \Omega$(n) $\wedge$ P $\in$ O(n^2)

Ejercicio 3: Editor de Texto

a) Escribir en castellano el invariante de representación

1. La pestaña seleccionada no puede estar en el conjunto de las pestañas vacías ni en el conjunto de las no vacías.
2. Las pestañas vacías no pueden estar en las no vacías y viceversa.
3. Todas las pestañas estan definidas en algún conjunto (vacías, no vacías o seleccionada).
4. Las pestañas no vacías no pueden tener su texto vacío.

b) Escribir formalmente el invariante de representación

Pasemos los predicados a lógica de primer orden.

(1) $e.seleccionada \notin e.inactivasVacías \wedge$
$(\forall t : tupla < nat, string >)(t \in e.inactivasNoVacías \Rightarrow \pi_1(t) \neq e.seleccionada)$

(2) $(\forall i : nat)(\forall t : tupla < nat, string >)((i \in e.inactivasVacías \wedge t \in e.inactivasNoVacías) \Rightarrow i \neq \pi_1(t))$

(3) $(\forall i : nat)(0 \leq i \leq long(e.inactivasVacías) + long(e.inactivasNoVacías)$
$\Rightarrow i = e.seleccionada \vee i \in e.inactivasVacías \vee$
$(\exists t : tupla < nat, string >)(t \in e.inactivasNoVacías \wedge \pi_1(t) = i))$

(4) $(\forall t : tupla < nat, string >)(t \in e.inactivasNoVacías \Rightarrow \neg vacía?(\pi_2(t)))$

Luego, nuestro invariante de representación queda de la forma

$$\text{Rep: } \widehat{estr} \to boolean$$
$$\text{Rep}(e) \equiv (1) \wedge (2) \wedge (3) \wedge (4)$$

c) Escribir formalmente la función de abstracción

$$\text{Abs: } \widehat{estr}\ e \to \text{Editor } \{\text{Rep}(e)\}$$
$$(\forall e : \widehat{estr})\ \text{Abs}(e) =_{obs} ed\text{: Editor } |$$
$$\#pestañas(ed) = long(e.inactivasVacías) + long(e.inactivasNoVacías) + 1 \wedge_L$$
$$(\forall i : nat)(0 \leq i \leq \#pestañas(ed) \Rightarrow_L seleccionada?(ed,\ i) = true \Leftrightarrow e.seleccionada = i) \wedge$$
$$(\forall i : nat)(i \in e.inactivasVacías \Rightarrow_L texto(ed, i) = <>) \wedge$$
$$(\forall t : tupla < nat, string >)(t \in e.inactivasNoVacías \Rightarrow_L texto(ed,\ \pi_1(t)) = \pi_2(t)) \wedge$$
$$texto(ed,\ e.seleccionada) = e.anteriores \circ e.posteriores \wedge_L$$
$$posicionCursor(ed) = long(e.anteriores)$$

- La cantidad de pestañas es la suma de las inactivas + 1.
- La seleccionada es la que esta en e.seleccionada.
- Los textos de las vacías son vacíos, los de las no vacías son su segundo valor de la tupla y el de la seleccionada es la suma de los anteriores con los posteriores.
- La posicion del cursor es la longitud de los caracteres anteriores.