

Análisis de Algoritmos Examen parcial

Rodolfo Conde

Octubre del 2020

1. (2pts) Para cada pareja de funciones $f, g: \mathbb{N} \rightarrow \mathbb{N}$, determina si $f = \Theta(g)$.
 - a) $f(n) = 100n$ y $g(n) = (\log n)^{100}$.
 - b) $f(n) = n^3$ y $g(n) = 2\sqrt[n]{n^3}$.
2. (8pts) Considera la siguiente descripción informal de un método para ordenar los elementos de un arreglo A : Primero encontramos el elemento más pequeño de A y lo intercambiamos con $A[1]$. Luego encontramos el segundo elemento más pequeño de A y lo intercambiamos con $A[2]$. Continuamos de esta forma con los primeros $n - 1$ elementos de A .
 - A) (2pts) Escribe pseudocódigo para implementar este proceso de ordenamiento (el OrdenamientoMisterioso).
 - B) (2pts) ¿Cuál es el invariante de ciclo que mantiene este algoritmo?
 - C) (2pts) Escribe la prueba formal de la correctez de tu algoritmo.
 - D) (2pts) Realiza el análisis formal de la complejidad de tiempo y espacio de tu algoritmo usando notación asintótica. Considera el mejor y el peor caso para el análisis.

1. (2pts) Para cada pareja de funciones $f, g: \mathbb{N} \rightarrow \mathbb{N}$, determina si $f = \Theta(g)$.

a) $f(n) = 100n$ y $g(n) = (\log n)^{100}$.

b) $f(n) = n^3$ y $g(n) = 2\sqrt{n^3}$.

a) $f(n) = 100n$, $g(n) = (\log n)^{100} \quad \not\Rightarrow f = \Theta(g)$

Determinar si existen $C_1, C_2 \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ $\not\Rightarrow \forall n > n_0$

$$C_1(\log n)^{100} \leq 100n \leq C_2(\log n)^{100}$$

$$C_1 \log n \leq (100n)^{\frac{1}{100}} \leq C_2 \log n$$

Usando la derivada en cada uno de los términos, podemos ver que tan rápidamente crece la función en términos de n

$$\frac{\partial C_1 \log n}{\partial n} = C_1 n^{-1} \quad \frac{\partial C_2 \log n}{\partial n} = C_2 n^{-1} \quad \frac{\partial (100n)^{\frac{1}{100}}}{\partial n} = (100n)^{-\frac{99}{100}}$$

$$\frac{C_1}{n} \leq \frac{1}{(100n)^{\frac{99}{100}}} \leq \frac{C_2}{n}$$

$$C_1 \leq (100n)^{\frac{1}{100}} \leq C_2$$

\therefore No hay C_1, C_2 tal que permita acotar su $(100n)^{\frac{1}{100}}$

$$b) f(n) = n^3 \text{ y } g(n) = 2^{\sqrt{n/3}}$$

$$b) f(n) = n^3 \quad g(n) = 2^{\sqrt{n/3}} \quad f = \Theta(g)$$

Determinar si existen $C_1, C_2 \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ s.t. $\forall n \geq n_0$

$$C_1 2^{\sqrt{n/3}} \leq n^3 \leq C_2 2^{\sqrt{n/3}}$$

$$C_1 \sqrt{n/3} \log 2 \leq 3 \log n \leq C_2 \sqrt{n/3} \log 2$$

$$\frac{\partial C_1 (\frac{n}{3})^{1/2}}{\partial n} = C_1 \frac{1}{2} \left(\frac{n}{3}\right)^{-1/2} \frac{1}{3} = C_1 \left(\frac{1}{6}\right) \left(\frac{n}{3}\right)^{-1/2} = C_1' n^{-1/2}$$

$$\frac{\partial 3 \log n}{\partial n} = \frac{3}{n} = 3 n^{-1}$$

$$\frac{\partial C_2 \left(\frac{n}{3}\right)^{1/2}}{\partial n} = C_2 \frac{1}{2} \left(\frac{n}{3}\right)^{-1/2} \frac{1}{3} = C_2' n^{-1/2}$$

$$C_1' n^{-1/2} \leq 3 n^{-1} \leq C_2' n^{-1/2}$$

$$C_1' \leq 3 n^{1/2} \leq C_2'$$

\therefore No podemos acotar con un número positivo cuando $n \rightarrow \infty$

$$C_1 \cancel{=} \frac{3}{\sqrt{n}}$$

2. (8pts) Considera la siguiente descripción informal de un método para ordenar los elementos de un arreglo A : Primero encontramos el elemento más pequeño de A y lo intercambiamos con $A[1]$. Luego encontramos el segundo elemento más pequeño de A y lo intercambiamos con $A[2]$. Continuamos de esta forma con los primeros $n - 1$ elementos de A .

- A) (2pts) Escribe pseudocódigo para implementar este proceso de ordenamiento (el OrdenamientoMisterioso).
- B) (2pts) ¿Cuál es el invariante de ciclo que mantiene este algoritmo?
- C) (2pts) Escribe la prueba formal de la correctez de tu algoritmo.
- D) (2pts) Realiza el análisis formal de la complejidad de tiempo y espacio de tu algoritmo usando notación asintótica. Considera el mejor y el peor caso para el análisis.

```
1 function Ordenamiento_misterioso(A)
2     n = length(A)
3     for i=0 to n-2 do
4         min_index = i
5         for j=i+1 to n-1 do
6             if A[j] < A[min_index] then
7                 min_index = j
8             end if
9         end for
10        swap (A[i], A[min_index])
11    end for
12    return A
13 end function
```

B) (2pts) ¿Cuál es el invariante de ciclo que mantiene este algoritmo?

La invariante de ciclo son la condición o condiciones que el algoritmo debe mantener al inicio, final y durante cada iteración.

For externo:

Para el ordenamiento misterioso, la invariante de ciclo es que los elementos del arreglo ordenado hasta el índice actual $A[0, \dots, i]$ contendrán los i elementos que sean más pequeños de todo el arreglo A . Adicionalmente estos elementos estarán ordenados

For interno

El segundo for (interno) también debe tener una invariante de ciclo que este acorde a la invariante de ciclo del for externo.

En este caso tenemos que antes que empiece cada iteración j del for, $A[\text{min-index}]$ será menor a cada elemento $A[i], \dots, A[j-1]$

C) (2pts) Escribe la prueba formal de la correctez de tu algoritmo.

El algoritmo termina ya que consiste de dos for que corren de $i+1$ a $n-1$ (interno) y de $i=0$ a $n-2$ (externo). En los loops se emplea solamente un if, una comparación, un swap y una asignación.

for interno:

Inicialización:

Antes de entrar a la primera iteración del loop $j=i+1$. Por lo tanto el subarreglo $A[i, \dots, j-1]$ es igual a $A[i]$. Como la linea 4 del algoritmo define $\text{min_index} = i$, tenemos que min_index es el elemento más pequeño (porque es el único elemento del arreglo $A[i, \dots, j-1]$) y por lo tanto el invariante se cumple.

Mantenimiento

Asumimos que antes de entrar a la j -ésima iteración es el índice del elemento más pequeño del subarreglo $A[i, \dots, j-1]$. Durante la j -ésima iteración se cumplirá uno de estos dos casos:

$$A[j] < A[\text{min_index}]$$

$$A[j] \geq A[\text{min_index}]$$

En el segundo caso la condición del if de la linea 6 no se cumple, por lo que no se ejecutará el código del if y min_index será el índice del elemento más pequeño del sub

arreglo $A[i, \dots, j]$

Por otra parte, en el primer caso, la linea 7 modifica el valor de min_index por el de j que corresponde a $A[j]$ y es mas pequeño. Como min_index corresponde al indice de un elemento menor o igual a los elementos del subarreglo

$A[i, \dots, j-1]$ entonces $A[j]$ es menor o igual que todos los elementos del subarreglo $A[i, \dots, j-1]$. Por lo tanto min_index será el indice del elemento más pequeño del subarreglo $A[i, \dots, j]$

Termino:

Por la condición de mantenimiento al termino del for interno, min_index será el indice del elemento más pequeño en el arreglo $A[i, \dots, n-1]$, ya que $j=n+1$ al finalizar la última iteración del for (ya nos "salimos").

De esta forma se habrá encontrado el elemento más pequeño del subarreglo $A[i, \dots, n-1]$.

For externo

Inicialización:

Al iniciar el for i es igual a 0 y $A[i]=A[0]$. El arreglo $A[0, \dots, 0]$ contiene un único elemento y por lo tanto está ordenado.

Mantenimiento:

Por el argumento del for interno sabemos que por la linea 9 $A[min_index]$ será el elemento más pequeño del subarreglo $A[i, \dots, n-1]$. En la linea 10 ejecutamos el intercambio entre $A[i]$ y $A[min_index]$. Al hacer esto tenemos que al pasar a la linea 11 el arreglo $A[0, \dots, i+1]$ contendrá los elementos más pequeños del arreglo original y que estos están ordenados. Esto sucede en todas las iteraciones.

Término:

El for termina cuando $i=n-2$. Por la condición de mantenimiento tenemos que los elementos del subarreglo $A[0, \dots, i+1]$ están ordenados y entre los elementos de $A[0, \dots, i] = A[0, \dots, n-1]$ estarán ordenados. Por lo tanto el algoritmo regresará A ordenado.

- D) (2pts) Realiza el análisis formal de la complejidad de tiempo y espacio de tu algoritmo usando notación asintótica. Considera el mejor y el peor caso para el análisis.

Space complexity: $O(1)$ [además del arreglo inicial A]

Utiliza un tamaño de memoria constante para las variables i, j, min-index sin importar el tamaño del arreglo a ordenar

Time complexity :

Al tener dos for anidados es señal que la complejidad de tiempo del algoritmo es $O(n^2)$, ya que tenemos que:

1	function Ordenamiento misterioso (A)	$C_1 = 1$
2	$n = \text{length}(A)$	$C_2 = 1$
3	for $i = 0$ to $n-2$ do	$C_3 = n-1$
4	$\text{min_index} = i$	$C_4 = n-1$
5	for $j = i+1$ to $n-1$ do	$C_5 = \sum_{j=1}^{n-1} (n-j)$
6	if $A[j] < A[\text{min_index}]$ then	$C_6 = \sum_{j=1}^{n-1} (n-j)$
7	$\text{min_index} = j$	$C_7 = \sum_{j=1}^{n-1} (n-j)$
8	end if	$C_8 = \sum_{j=1}^{n-1} (n-j)$
9	end for	$C_9 = (n-1)$
10	swap ($A[i], A[\text{min_index}]$)	$C_{10} = (n-1)$
11	end for	$C_{11} = (n-1)$
12	return A	$C_{12} = 1$
13	end function	$C_{13} = 1$

$$C_1 + C_2 + C_{12} + C_{13} + (C_3 + C_4 + C_9 + C_{10} + C_n)(n-1) + (C_5 + C_6 + C_7 + C_8) \sum_{j=1}^{n-1} (n-j)$$

Tenemos que cuando:

$i=0$ el for interno corre de $j=1, \dots, n-1$ es decir $n-1$ veces

$i=1$ " " " $j=2, \dots, n-1$ " " $n-2$ veces

:

$i=(n-2)$ " " " $j=n-1, \dots, n-1$ 1 vez

$$(n-1) + (n-2) + \dots + 1 = \frac{n(n-1)}{2} = \frac{n^2-n}{2} \text{ que es } O(n^2)$$