

## ACTIVIDAD 4: RESOLVER EJERCICIOS PROPUESTOS (COMO ACTIVIDAD ASÍNCRONA)

1. Calcular la complejidad de las siguientes ecuaciones de recurrencia por reducción:

1.  $T(n) \leq n + T(n-1); T(0) = c$
2.  $T(n) \leq c + 2T(n-1); T(1) = c$
3.  $T(n) \leq n + T(n/2); T(1) = c$
4.  $T(n) \leq c + 2T(n/2); T(1) = c$

$$1.- \quad T(n) \leq \begin{cases} c & , n=0 \\ 1+T(n-1)+n & , \text{otro caso} \end{cases}$$

Notamos  $a=1, b=1, K=1$  ya que  $O(n^k) = O(n) = n$   
 $\Rightarrow O(n^{k+1}) = O(n^2)$  orden cuadrático

$$2.- \quad T(n) \leq \begin{cases} c & , n=1 \\ 2+T(n-1)+c & , \text{otro caso} \end{cases}$$

orden exponencial

Notamos  $a=2, b=1, K=0 \Rightarrow O(a^k) = O(2^n)$

$$3.- \quad T(n) \leq \begin{cases} c & , n=1 \\ 1+T(n/2)+n & , \text{otro caso} \end{cases}$$

Notamos  $a=1, b=2, K=1$ , como  $a < b^K \Rightarrow 1 < 2$   
 $\Rightarrow O(n^k) = O(n)$  orden lineal

$$4.- \quad T(n) \leq \begin{cases} c & , n=1 \\ 2+T(n/2)+c & , \text{otro caso} \end{cases}$$

Notamos  $a=2, b=2, K=0$ , como  $a > b^K \Rightarrow 2 > 1$   
 $\Rightarrow O(n^{\log_b(a)}) = O(n^{\log_2 2}) = O(n)$  orden lineal

2. Para un problema p se conocen dos algoritmos que lo resuelven. Se sabe que el tiempo que tarda el primer algoritmo en resolver una instancia de tamaño n es  $T(n) = 32n + 5$ , mientras que el segundo tarda  $T(n) = 2n^2 + 3$ . En general, ¿cuál de los dos algoritmos es más eficiente? ¿Por qué? ¿Existe un valor crítico de n tal que, para instancias menores, el mejor algoritmo sea otro?

El algoritmo más eficiente es el primero, porque es de orden  $O(n)$  y el segundo de  $O(n^2)$ , ( $O(n^2) > O(n)$ ). ~~valor crítico 1 (P)~~

3. Una joven pareja de conejos es colocada en una isla. Un par de conejos no se reproduce hasta que tiene 2 meses de edad. Despues que estos tienen los 2 meses de edad, cada par produce un nuevo par. Se pide encontrar una relación de recurrencia para el número de pares de conejos en la isla al cabo de  $n$  meses, asumiendo que ningún conejo muere. A partir de dicha relación de recurrencia, se debe diseñar un algoritmo para determinar la cantidad de conejos existente en un momento dado. Plantear el  $T(n)$  para dicho algoritmo y calcular su  $O()$  de complejidad.

## Ejercicio de Métodos de Programación

1 P

1M



1P

2M



1P

$\rightarrow 1+0$

3M



2P

4M



3P

5M



4P

6M



5P

Conejos ( $n$  mes  $m$ ):  $M_{nm}$

si:  $n = 0 \quad o \quad n = 1$  Entonces  
devolver (1)

3c

c

Valor  $\leftarrow$  Conejos( $m-1$ ) + Conejos( $m-2$ )

4c

devolver (Valor)

c

$$T(n) = \begin{cases} Sc & n=0, n=1 \\ T(n-1) + T(n-2) + Sc, & \text{otro caso} \end{cases}$$

$$\Rightarrow T(n) \leq 2 \cdot T(n-1) + O(1)$$

$$O(\alpha^{n/b}) = O(2^n)$$

$$\alpha = 2 \\ b = 1 \\ k = 0$$

orden  
exponencial

4. Se dispone de un arreglo A que contiene  $n$  números enteros desordenados, que pueden ser positivos o negativos. Se requiere escribir un algoritmo que reordene estos datos dejando alternados valores positivos y negativos, comenzando con uno positivo. Si hay más de un tipo que del otro, los que sobran deben quedar al extremo izquierdo del arreglo. El proceso debe demorar a lo más tiempo proporcional al tamaño del arreglo ( $O(n)$ ) y debe efectuarse usando solamente el arreglo A entregado, sin usar otro arreglo auxiliar. Justificar complejidad lineal del algoritmo propuesto.

	1	2	3	4	5	No	3	cero				
ANTES:	-15	17	42	60	-32	21	13	-44	-72	24	-25	37
DESPUÉS:	17	-15	42	-32	60	-44	21	-72	13	-25	24	37

Algo (array A): array

valor  $\leftarrow 0$ , contPos  $\leftarrow 0$ , contNeg  $\leftarrow 0$

$n \leftarrow$  largo array (A)

$a \leftarrow 1$

contar las  
cantidad  
de positivos  
y negativos

Para  $i \leftarrow 1$  hasta  $n$   
si:  $A[i] > 0$  hacer  
contPos  $\leftarrow$  contPos + 1

sino

contNeg  $\leftarrow$  contNeg + 1

contRealPos  $\leftarrow 0$ , contRealNeg  $\leftarrow 0$

primero ordenar los  
 $Z^+$  en indices impares

para  $i \leftarrow 1$  hasta  $n$

valor  $\leftarrow A[i]$

Mientras contRealPos  $<$  contPos      hacer  
si:  $a \% 2 \neq 0$  Entonces  
indice  $\rightarrow Z^+$

si: valor  $> 0$  Entonces

$A[i] \leftarrow A[a]$

$A[a] \leftarrow$  valor

contRealPos  $\leftarrow$  contRealPos + 1

sino      valor  $\leftarrow 0$

$a \leftarrow a - 1$

sino

$i \leftarrow i - 1$

$a \leftarrow a + 1$

ordenar

los

$Z^+$   
en los  
indices

ímpar

ORdenar  
 los  
 $\geq$   
 en los  
 indices  
 pél

$\rightarrow$  los negativos van desde la posic 2

```

    a ← 2
    Para i ← 1 hasta n
      valor ← A[i]
      Mientras contRecNeg < contNeg hacer
        si: a % 2 = 0 entonces
          si: valor < 0 entonces
            A[i] ← A[a]
            A[a] ← valor
            contRecNeg ← contRecNeg + 1
        sino
          a ← a - 1
        sino
          i ← i - 1
          a ← a + 1
  
```

Devolver (A)

\* Este algoritmo es  $O(n)$   
 NO sé si es eficaz, luego lo veré.

También puede hacer bucles anidados pero así la complejidad sería  $O(n^2)$ , y pedían que fuera  $O(n)$ .

Ojalá esté bien aunque lo dudo.

5. Ordenar en orden creciente cada uno de los siguientes conjuntos de funciones de  $T(n)$  de diferentes algoritmos, indicando el valor de  $n$  para el que se cumple el orden indicado.

a)

$$\begin{array}{l} \sqrt{n} \\ n \log n \\ n^2 \\ n^{\frac{1}{3}} + \log n \\ \ln n \\ (1/3)^n \end{array}$$

b)

$$\begin{array}{l} n \\ n - n^3 + 7n^5 \\ n^3 \\ (\log n)^2 \\ \frac{n}{\log n} \\ (3/2)^n \end{array}$$

c)

$$\begin{array}{l} 2^n \\ n^2 + \log n \\ \log n \\ n! \\ \log \log n \\ 6 \end{array}$$