

 Universidad AUTÓNOMA de Occidente	UNIVERSIDAD AUTÓNOMA DE OCCIDENTE					Valoración
	FACULTAD DE INGENIERÍA NUCLEO MIDIA			ESTRUCTURAS DE DATOS Y ALGORITMOS 1	GRUPO 1 HL	
	CÓDIGO:		NOMBRE:			
EXAMEN PARCIAL - CORTE 2						FECHA: Abril 15 de 2024

PRIMERA PARTE	Tiempo de ejecución y complejidades [3 Puntos]	PUNTAJE	
---------------	--	---------	--

- a) [1.0 pts] A partir del siguiente fragmento en el que se han definido los bloques que permiten realizar su análisis, n que es igual al tamaño del arreglo *datos*, indicar: 1) cuantas veces se ejecuta la condición **n1**
2) indicar cuantas veces se ejecuta la instrucción **n2** en el *peor de los casos*.

```

                                n1
for ( let i=7 ; i < n+1 ; i++ ) {
    if ( datos[i-3] < datos[0] ){
n2 ⇒ aux = datos[i-4];
    }else{
        veces += ajustarSuma(datos);
    }
}

```

R://

1)

2)

- b) [0.8 pts] A partir de las siguientes funciones $T_A(n)$ y $T_B(n)$, correspondientes, respectivamente, al número total de instrucciones, de los algoritmo A y B. a) indicar sus órdenes de complejidad b) indicar cuál de los algoritmos es más eficiente, con base las ordenes de ejecución obtenidos:

$$T_A(n) = 5*n^2 + 1000*n + 50000, \quad T_B(n) = 0.1*n^2 + 0.0001*n^3 - 20$$

R://

$$T_A(n) = O(\quad), \quad T_B(n) = O(\quad)$$

El algoritmo más eficiente es el ____ (indicar la letra)

- c) [1.2 pts] A partir del siguiente fragmento, en el cual n es el tamaño del arreglo *datos* (obtenido con la instrucción *datos.length*), la complejidad del método *calcularMetodo1* es $O(\lg n)$ y la del método *calcularMetodo2* es $O(n)$, indicar las complejidades O de: a) todas las ejecuciones de las líneas solicitadas b) de todo el fragmento.

```

27 let valA = datos.length;
28 let valB = Math.trunc(Math.sqrt(valA));
29 for ( let i = 0 ; i < valA ; i++ ) {
30     for ( let j=1 ; j<valB ; j++ ){
31         res = calcularMetodo1(datos,j);
32         suma += calcularMetodo2(datos);
33     }
34 }

```

R: //

Línea	Respuesta a)	Respuesta b)
27 y 28	$T(n) = O(1)$	
29		
30		
31 y 32		

SEGUNDA PARTE	Recursividad [2.0 Puntos]	PUNTAJE	
---------------	---------------------------	---------	--

A partir de un algoritmo A, del cual se muestra su pseudocódigo y relación de recurrencia $T_A(n)$ que define su tiempo de ejecución:

```

PROCEDIMIENTO solucionG1 (A,n)
    SI n es igual a 1
        retornar la posicion obtenida al buscar en A el valor 500
    SINO
        crear variable res y asignar el valor obtenido de buscar el numero mayor de A
        EJECUTAR cuatro veces en un ciclo
            acumular en res el valor obtenido de invocar a solucionG1 con un cuarto del
            valor de n
        retornar res

```

$$T_A(n) = \begin{cases} O(n) & n = 1 \\ O(n) + 4T(\frac{n}{4}) & n > 1 \end{cases}$$

- a) [2.0 pts] Calcular su complejidad (realizando sustituciones o generando el árbol) y comparar su eficiencia contra la de un algoritmo B. Se requiere:
- expandir $T_A(n)$ k veces e indicarla en términos de k
 - despejar k (cuantas veces se realizan las invocaciones recursivas)
 - despejar $T_A(n)$ al reemplazar el valor de k

IV) indicar la complejidad O del algoritmo A

V) si la complejidad de un algoritmo B que realiza la misma labor es $O(n^3)$, indicar cuál de los dos algoritmos (A o B) es más eficiente.

R://

I) [0.8 pts]) expandir $T_A(n)$ k veces e indicarla en términos de k

$$T_A(n) =$$

II) [0.4 pts] despejar k (cuantas veces se realizan las invocaciones recursivas)

III) [0.3 pts] despejar $T_A(n)$ al reemplazar el valor de k

$$T_A(n) =$$

IV) [0.3 pts] indicar la complejidad O del algoritmo A

$$T_A(n) =$$

V) [0.2 pts] indicar cual algoritmo es más eficiente

El algoritmo más eficiente es el ____.

