COMPLEJIDAD: EJERCICIOS

N	ENUNCIADO
1	¿Cuál es el objetivo de la etapa de análisis en el Diseño y Análisis de un Algoritmo?: a. Determinar el lenguaje y herramientas disponibles para su desarrollo. b. Estimar los recursos que consumirá el algoritmo una vez implementado. c. Estimar la potencia y características del equipo informático necesarios para el correcto funcionamiento del algoritmo.
2	¿Cuál de las siguientes jerarquías de complejidades es la correcta? a. $O(1) \subset O(\lg n) \subset O(\lg \lg n) \subset \ldots$ b. $\ldots \subset (n!) \subset O(2^n) \subset O(n^n)$ c. $\ldots \subset (2^n) \subset O(n!) \subset O(n^n)$
3	¿Cuál de los siguientes algoritmos de ordenación tiene menor complejidad? a. Burbuja b. Inserción directa c. Mergesort
4	¿El tiempo de ejecución de un algoritmo depende de la talla del problema? a. Sí, siempre b. No, nunca c. No necesariamente
5	Ordena de menor a mayor las siguientes complejidades 1. O(1) 2. O(n^2) 3. O(nlgn) 4. O(n!) a. 3,1,2 y 4 b. 1,3,2 y 4 c. 1,3,4 y 2
6	El estudio de la complejidad resulta realmente interesante para tamaños grandes de problema por varios motivos: a. Las diferencias reales en tiempo de compilación de algoritmos con diferente coste para tamaños pequeños del problema no suelen ser muy significativas. b. Las diferencias reales en tiempo de ejecución de algoritmos con diferente coste para tamaños grandes del problema no suelen ser muy significativas. c. Ninguna de las anteriores.
7	¿Por que se emplean funciones de coste para expresar el coste de una algoritmo? a. Para poder expresar el coste de los algoritmos con mayor exactitud b. Para que la expresión del coste del algoritmo sea válida para cualquier entrada al mismo c. Para poder expresar el coste de un algoritmo mediante una expresión matemática
8	El caso base de una ecuación de recurrencia asociada a la complejidad temporal de un algoritmo expresa: a. El coste de dicho algoritmo en el mejor de los casos. b. El coste de dicho algoritmo en el peor de los casos. c. Ninguna de las anteriores
9	La complejidad de la función TB es: función TB (A: vector[λ]; iz , de : *N)* : *N* var n,i:N; n=iz-de+1 opcion (n < 1) : devuelve (0) ; (n = 1) : devuelve (1) ; (n > 1) : si (A[iz] = A[de]) entonces devuelve (TB(A, iz + 1, de – 1) + 1); sino devuelve (TB(A, iz + 1, de – 1)) ; finsi ; fopcion fin a. $\Theta(n)$ b. $\Theta(n \cdot \lg n)$ c. $\Theta(n^2 \cdot \lg n)$
10	Dado el polinomio $f(n)= a_m n^m + a_{m-1} n^{m-1} + \ldots + a_0$, con $a_m \in R^+$ entonces f pertenece al orden: a. $O(n^m)$. b. $\Omega(n^m)$. c. La dos respuestas anteriores son correctas.
11	Si $f1(n) \in O(g1(n))$ y $f2(n) \in O(g2(n))$ entonces: a. $f1(n) \cdot f2(n) \in O(\text{maximo}(g1(n), g2(n)))$ b. $f1(n) \cdot f2(n) \in O(g1(n) \cdot g2(n))$ c. Ambas son correctas
12	Si $f1(n) \in O(g1(n))$ y $f2(n) \in O(g2(n))$ entonces: a. $f1(n)+f2(n) \in O(\text{maximo}(g1(n), g2(n)))$ b. $f1(n)+f2(n) \in O(g1(n)+g2(n))$ c. Ambas son correctas
13	Un algoritmo cuya talla es n y que tarda 40^n segundos en resolver cualquier instancia tiene una complejidad temporal: a. $\Theta(n^n)$ b. $\Theta(4^n)$ c. Ninguna de las anteriores

14 Si dos algoritmos tienen la misma complejidad asintótica:

a. No necesitan exactamente el mismo tiempo para su ejecución.
b. Necesitan exactamente el mismo tiempo para su ejecución.
c. Ninguna de las anteriores

15 Los algoritmos directos de ordenación, respecto de los indirectos:

a. Presentan una mayor complejidad temporal y sus tiempos de ejecución absolutos son mayores.
b. Presentan una menor complejidad temporal y sus tiempos de ejecución absolutos son menores.
c. Presentan una mayor complejidad temporal si bien sus tiempos de ejecución absolutos son menores.

16 La talla o tamaño de un problema depende de:

a. Conjunto de valores asociados a la entrada y salida del problema.
b. Conjunto de valores asociados a la salida del problema.
c. Conjunto de valores asociados a la entrada del problema.

17 En un algoritmo recursivo, la forma de dividir el problema en subproblemas:

a. Influye en la complejidad espacial del mismo.
b. Influye en su complejidad temporal.
c. No influye en ninguna de sus complejidades.

18 f(n) = 5n+3m·n +11 entonces f(n) pertenece a:

a. O (n·m).
b. O (n^m).
c. Las dos son correctas

19 El sumatorio, desde i=1 hasta n, de i^k pertenece a:

a. $O(n^{k+1})$
b. $O(n^k)$
c. Ninguna de las anteriores

20 La complejidad de la función A2 es:

```
Funcion A2 (n, a: entero):entero;
Var  r: entero; fvar
   si (a^2 > n) devuelve 0
   sino
       r:= A2(n, 2a);
       opción
            n < a^2:      devuelve r;
            n ≥ a^2:      devuelve r + a;
        fopción
   fsi
   fin
```

a. $O(\sqrt{n} \cdot a)$
b. $O(\sqrt{n} / a)$
c. $O(n / \sqrt{a})$

21 Cual de las siguientes definiciones es cierta:

a. Las cotas de complejidad se emplean cuando para una misma talla se obtienen diferentes complejidades dependiendo de la entrada al problema.
b. Las cotas de complejidad se emplean cuando para diferentes tallas se obtienen diferentes complejidades dependiendo de la entrada al problema.
c. Ninguna de las anteriores

22 Cuando para distintas instancias de problema con el mismo tamaño no obtenemos el mismo resultado:

a. No es posible calcular la complejidad a priori y debemos ejecutar el programa varias veces con la misma talla y obtener el tiempo medio para hallar la complejidad media.
b. No se puede aplicar la técnica de paso de programa, ya que esta técnica es para calcular la complejidad a priori.
c. Calculamos el máximo y mínimo coste que nos puede dar el algoritmo.

23 f(n) = 5n+5 ¿ f(n) pertenece a O(n)?

a. Si. El valor de *c* es 5 y el valor mínimo de n_0 es de 3
b. Si. El valor de *c* es 9 y el valor mínimo de n_0 es de 1
c. Si. El valor de *c* es 6 y el valor mínimo de n_0 es de 5

24 f(n) = 10n+7 ¿ f(n) pertenece a $O(n^2)$?

a. Si. Para c = 1 y a partir de un valor de n_0 =10.
b. Sí Para cualquier valor de c positivo siempre existe un n_0 a partir del que se cumple.
c. No.

25 Si f(n) ∈ Ω (g(n)) entonces:

a. $\exists$ c, $n_0 \in R^+$: $f(n) \geq c \cdot g(n)$ $\forall$ $n \geq n_0$
b. $\exists$ c, $n_0 \in R^+$: $f(n) \geq c \cdot g(n)$ $\forall$ n
c. $\exists$ c, $n_0 \in R^+$: $f(n) \leq c \cdot g(n)$ $\forall$ $n \geq n_0$

26 El coste asociado a la siguiente ecuación de recurrencia es:

$$f(n)\begin{cases}1 & n \leq 1 \\ n + f(n/2) + f(n/2) & n > 1\end{cases}$$

a. $\Theta(n \lg n^2)$
b. $\Theta(n^2 \lg n)$
c. $\Theta(n \lg n)$