Respuestas para la modalidad **3**

1. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T(\frac{n}{2}) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in O(n)$, ¿en cuál de estos tres casos nos podemos encontrar?

(a) $g(n) = \log n$
$\boxed{\text{(b)}}$ $g(n) = 1$
(c) $g(n) = n$

2. ¿Cuál de estos problemas tiene una solución eficiente utilizando *programación dinámica*?

(a) El problema de la asignación de tareas.
(b) La mochila discreta sin restricciones adicionales.
$\boxed{\text{(c)}}$ El problema del cambio (devolver una cantidad de dinero con el mínimo número de monedas).

3. En los algoritmos de *backtracking*, ¿Puede el valor de una cota pesimista ser mayor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)

(a) No, el valor de la cota pesimista de un nodo nunca puede ser superior al de la cota optimista de ese mismo nodo.
(b) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
$\boxed{\text{(c)}}$ En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

4. ¿Para qué puede servir la cota pesimista de un nodo de *ramificación y poda*?

$\boxed{\text{(a)}}$ Para actualizar el valor de la mejor solución hasta el momento.
(b) Para descartar el nodo si no es prometedor.
(c) Para obtener una cota optimista más precisa.

5. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.

(a) $\Theta(\log^2(n)) = \Theta(\log^3(n))$
(b) $\log(n^3) \notin \Theta(\log_3(n))$
$\boxed{\text{(c)}}$ $\Theta(\log_2(n)) = \Theta(\log_3(n))$

6. Dada la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT(\frac{n}{a}) + g(n) & \text{en otro caso} \end{cases}$$

(donde p y a son enteros mayores que 1 y $g(n) = n^k$), ¿qué tiene que ocurrir para que se cumpla $T(n) \in \Theta(n^k)$?

$\boxed{\text{(a)}}$ $p < a^k$

(b) $p > a^k$

(c) $p = a^k$

7. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T(\frac{n}{2}) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es *cierta*:

(a) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de de ordenación *quicksort*.

$\boxed{\text{(b)}}$ Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de de ordenación *quicksort*.

(c) Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de de ordenación *mergesort*.

8. ¿Cuál es la complejidad temporal de la siguiente función?

```
unsigned examen (unsigned n) {
unsigned i=n, k=0;
   while (i>0){
       unsigned j=i;
       do{
          j = j * 2;
          k = k + 1;
       } while (j<=n);
       i = i / 2;
   }
   return k;
}
```

(a) $\Theta(\log n)$

$\boxed{\text{(b)}}$ $\Theta(\log^2 n)$

(c) $\Theta(n)$

9. ¿Cuál es la complejidad temporal en el mejor de los casos de la siguiente función?

```
void examen (vector <int >& v){
int i=0,j,x,n=v.size();
bool permuta=1;
    while (n>0 && permuta){
       i=i+1;
       permuta=0;
       for (j=n-1;j>=i;j--)
          if (v[j] <v[j-1]){
             x=v[j];
             permuta=1;
             v[j]=v[j-1];
             v[j-1]=x;
          }
    }
}
```

(a) $\Omega(1)$
$\boxed{\text{(b)}}$ $\Omega(n)$
(c) Esta función no tiene caso mejor.

10. Ante un problema de optimización resuelto mediante *backtracking*, ¿Puede ocurrir que el uso de las cotas pesimistas y optimistas sea inútil, incluso perjudicial?

$\boxed{\text{(a)}}$ Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.
(b) Según el tipo de cota, las pesimistas puede que no descarten ningún nodo pero el uso de cotas optimistas garantiza la reducción el espacio de búsqueda.
(c) No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por tanto la eficiencia del algoritmo.

11. ¿Cuál de estas afirmaciones es *cierta*?

$\boxed{\text{(a)}}$ La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.
(b) La ventaja de la solución de programación dinámica iterativa al problema de la mochila discreta es que nunca se realizan cálculos innecesarios.
(c) Los algoritmos iterativos de programación dinámica utilizan memoización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar.

12. ¿Para cuál de estos problemas de optimización se conoce al menos una solución voraz óptima?

    $\boxed{\text{(a)}}$ El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.

    (b) El problema de la mochila discreta.

    (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j, c_{ij} está tabulado en una matriz.

13. Garantiza el uso de una estrategia "divide y vencerás" la existencia de una solución de complejidad temporal polinómica a cualquier problema?

    $\boxed{\text{(a)}}$ No.

    (b) Sí, en cualquier caso.

    (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.

14. La complejidad temporal del la solución de *vuelta atrás* al problema de la mochila discreta es ...

    $\boxed{\text{(a)}}$ ...exponencial en el caso peor..

    (b) ...cuadrática en el caso peor.

    (c) ...exponencial en cualquier caso.

15. Cuál de los siguientes criterios proporcionaría una cota optimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).

    (a) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.

    $\boxed{\text{(b)}}$ Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.

    (c) Utilizar la solución (subóptima) que se obtiene al resolver el problema mediante un algoritmo voraz.

16. Se quiere reducir la complejidad temporal de la siguiente función haciendo uso de programación dinámica. ¿Cuál sería la complejidad temporal resultante?

```
unsigned g( unsigned n, unsigned r){
   if (r==0 || r==n)
      return 1;
   return g(n-1, r-1) + g(n-1, r);
}
```

    (a) Se puede reducir hasta lineal.

    $\boxed{\text{(b)}}$ Cuadrática

    (c) La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.

17. Si $\lim_{n\to\infty}(f(n)/n^2) = k$, y $k \neq 0$, ¿cuál de estas tres afirmaciones es *cierta*?

$\boxed{\text{(a)}}$ $f(n) \in \Theta(n^2)$

(b) $f(n) \in \Omega(n^3)$

(c) $f(n) \in \Theta(n^3)$

18. Sea A una matriz cuadrada $n \times n$. Se trata de buscar una permutación de las columnas tal que la suma de los elementos de la diagonal de la matriz resultante sea mínima. Indicad cuál de las siguientes afirmaciones es correcta.

(a) La complejidad temporal de la mejor solución posible al problema es $O(n \log n)$.

(b) La complejidad temporal de la mejor solución posible al problema está en $\Omega(n^n)$.

$\boxed{\text{(c)}}$ Si se construye una solución al problema basada en el esquema de ramificación y poda, una buena elección de cotas optimistas y pesimistas podría evitar la exploración de todas las permutaciones posibles.

19. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.

$\boxed{\text{(a)}}$ $\Theta(f) = O(f) \cap \Omega(f)$

(b) $O(f) = \Omega(f) \cap \Theta(f)$

(c) $\Omega(f) = \Theta(f) \cap O(f)$

20. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.

$\boxed{\text{(a)}}$ $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$

(b) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$

(c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

21. Cuando se resuelve el problema de la mochila discreta usando la estrategia de vuelta atrás, ¿puede ocurrir que se tarde menos en encontrar la solución óptima si se prueba primero a meter cada objeto antes de no meterlo?

$\boxed{\text{(a)}}$ Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.

(b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.

(c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

22. En los algoritmos de *ramificación y poda* ...
    - $\boxed{\text{(a)}}$ Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.
    - (b) Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
    - (c) Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.

23. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...
    - (a) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
    - (b) ... determina la complejidad temporal en el peor de los casos del algoritmo.
    - $\boxed{\text{(c)}}$ ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.

24. Los algoritmos de *vuelta atrás* que hacen uso de cotas optimistas generan las soluciones posibles al problema mediante ...
    - (a) ... un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
    - $\boxed{\text{(b)}}$ ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
    - (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

25. ¿Qué se entiende por *tamaño del problema*?
    - $\boxed{\text{(a)}}$ La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.
    - (b) El valor máximo que puede tomar una instancia cualquiera de ese problema.
    - (c) El número de parámetros que componen el problema.

26. El algoritmo de ordenación *Quicksort* divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?
    - (a) $O(1)$
    - $\boxed{\text{(b)}}$ $O(n)$
    - (c) $O(n \log n)$

27. Se quiere ordenar d números distintos comprendidos entre 1 y n. Para ello se usa un array de n booleanos que se inicializan primero a *false*. A continuación se recorren los d números cambiando los valores del elemento del vector de booleanos correspondiente a su número a *true*. Por último se recorre el vector de booleanos escribiendo los índices de los elementos del vector de booleanos que son *true*. ¿Es este algoritmo más rápido (asintóticamente) que el *mergesort*?

    $\boxed{\text{(a)}}$ Sólo si $d \log d > kn$ (donde k es una constante que depende de la implementación)

    (b) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$

    (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

28. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, ¿De qué manera se debería ordenar la lista de nodos vivos para obtener una solución aceptable?

    (a) Por cota optimista: explorando primero los nodos con menor cota optimista.

    (b) Por cota pesimista: explorando primero los nodos con menor cota pesimista.

    $\boxed{\text{(c)}}$ las otras dos opciones pueden ser ambas correctas.

29. ¿Cuál es la definición correcta de $\Omega(g)$?

    $\boxed{\text{(a)}}$ $\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$

    (b) $\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$

    (c) $\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$

30. La solución recursiva ingenua (pero correcta) a un problema de optimización llama más de una vez a la función con los mismos parámetros. Una de las siguientes tres afirmaciones es falsa.

    $\boxed{\text{(a)}}$ Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.

    (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.

    (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

31. En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento...
    - $\boxed{\text{(a)}}$ ... pueden eliminar soluciones parciales que son factibles.
    - (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
    - (c) Las otras dos opciones son ambas son verdaderas.

32. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, una cota optimista es el resultado de asumir que ...
    - $\boxed{\text{(a)}}$ ... no se van a utilizar colores distintos a los ya utilizados.
    - (b) ... se van a utilizar tantos colores distintos a los ya utilizados como vértices quedan por colorear.
    - (c) ... sólo va a ser necesario un color más.

33. Supongamos el algoritmo de ordenación *Mergesort* modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres. Posteriormente se combinan las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?
    - (a) Ninguna de las otras dos opciones es cierta.
    - $\boxed{\text{(b)}}$ $\Theta(n)$
    - (c) $\Theta(\log_3 n)$

34. Se quiere reducir la complejidad temporal de la función `g` usando programación dinámica iterativa. ¿cuál sería la complejidad espacial del algoritmo resultante?

```
int g( int p[], unsigned n ) {
   if (n==0)
      return 0;
   int q = -1;
   for ( unsigned i = 1; i <= n; i++ )
      q = max( q, p[i] + g( p, n-i ) );
   return q;
}
```

    - (a) Cuadrática.
    - $\boxed{\text{(b)}}$ Lineal.
    - (c) Cúbica.

35. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ entonces
    - (a) $f \notin \Theta(\text{máx}(g_1, g_2))$
    - (b) $f \in \Theta(g_1 \cdot g_2)$
    - $\boxed{\text{(c)}}$ $f \in \Theta(g_1 + g_2)$