

1. Dado un problema de optimización cualquiera, ¿la estrategia de *vuelta atrás* garantiza la solución óptima?
 - (a) Sí, puesto que ese método analiza todas las posibilidades.
 - (b) Sí, siempre que el dominio de las decisiones sea discreto o discretizable y además se empleen mecanismos de poda basados en la mejor solución hasta el momento.
 - (c) Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
2. En los algoritmos de *ramificación y poda* ...
 - (a) Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
 - (b) Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.
 - (c) Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.
3. 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.
 - (a) 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.
 - (b) 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.
 - (c) Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
4. Si un problema de optimización lo es para una función que toma valores continuos ...
 - (a) La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
 - (b) La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
 - (c) El uso de memoria de la programación dinámica iterativa y de la programación dinámica recursiva es el mismo independientemente de si el dominio es discreto o continuo.

5. El uso de funciones de cota en ramificación y poda...
- (a) ... transforma en polinómicas complejidades que antes eran exponenciales.
 - (b) ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.
 - (c) ... puede reducir el número de instancias del problema que pertenecen al caso peor.
6. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?
- (a) Se multiplica k por la distancia de la arista más corta que nos queda por considerar, donde k es el número de saltos que nos quedan por dar.
 - (b) Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
 - (c) Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las k aristas más cortas, donde k es el número de saltos que nos quedan por dar.
7. ¿Para cuál de estos problemas de optimización existe una solución voraz?
- (a) El problema de la mochila discreta.
 - (b) 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.
 - (c) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
8. Se desea encontrar el camino más corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geométrica (en línea recta) entre cada par de ciudades. Se pretende acelerar la búsqueda de un algoritmo de *ramificación y poda* priorizando los nodos vivos (ciudades) que estén a menor distancia geográfica de la ciudad objetivo.
- (a) El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
 - (b) El nuevo algoritmo siempre será más rápido.
 - (c) Esta estrategia no asegura que se obtenga el camino más corto.

9. La complejidad temporal en el mejor de los casos...

- (a) ... es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar.
- (b) Las otras dos opciones son ciertas.
- (c) ... es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.

10. La mejor solución que se conoce para el problema de la mochila continua sigue el esquema ...

- (a) ... divide y vencerás.
- (b) ... ramificación y poda.
- (c) ... voraz.

11. La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda* ...

- (a) ... es siempre exponencial con el número de decisiones a tomar.
- (b) ... puede ser polinómica con el número de decisiones a tomar.
- (c) ... suele ser polinómica con el número de alternativas por cada decisión.

12. Cuando se usa un algoritmo voraz para abordar la resolución de un problema de optimización por selección discreta (es decir, un problema para el cual la solución consiste en encontrar un subconjunto del conjunto de elementos que optimiza una determinada función), ¿cuál de estas tres cosas es imposible que ocurra?

- (a) Que el algoritmo no encuentre ninguna solución.
- (b) Que se reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.
- (c) Que la solución no sea la óptima.

13. Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

- (a) Programación dinámica.
- (b) Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- (c) El método voraz

14. Sea la siguiente relación de recurrencia

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

Si $T(n) \in O(n^2)$, ¿en cuál de estos tres casos nos podemos encontrar?

- (a) $g(n) = n$
- (b) $g(n) = n^2$
- (c) $g(n) = 1$

15. Un algoritmo recursivo basado en el esquema *divide y vencerás* ...
- (a) ... nunca tendrá una complejidad exponencial.
 - (b) ... será más eficiente cuanto más equitativa sea la división en subproblemas.
 - (c) Las dos anteriores son ciertas.
16. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
 - (b) Las otras dos opciones son ciertas.
 - (c) ... pueden eliminar soluciones parciales que son factibles.
17. Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica
- (a) El problema de la mochila discreta.
 - (b) El problema de las torres de Hanoi
 - (c) El problema de cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n de manera que se maximice el precio de acuerdo con una tabla que da el precio para cada longitud.
18. El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...
- (a) ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
 - (b) ... una cota superior para el valor óptimo.
 - (c) ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
19. Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente
 - (b) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido
 - (c) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos
20. Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en otro de tamaño $n - 1$. Por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante. ¿cuál de estas clases de coste temporal asintótico es la más ajustada?
- (a) $O(2^n)$
 - (b) $O(n^3)$
 - (c) $O(n^2)$

21. El siguiente programa resuelve el problema de cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n de manera que se maximice el precio de acuerdo con una tabla que da el precio para cada longitud, pero falta un trozo. ¿Qué debería ir en lugar de XXXXXXX?

```
void fill(price r[]) {  
    for (index i=0;i<=n;i++) r[i]=-1;  
}  
  
price cutrod(price p[], r[], length n) {  
    price q;  
    if (r[n]>=0) return r[n];  
    if (n==0) q=0;  
    else {  
        q=-1;  
        for (index i=1;i<=n;i++)  
            q=max(q,p[i]+cutrod(XXXXXXX));  
    }  
    r[n]=q;  
    return q;  
}
```

- (a) $p, r-1, n$
(b) $p, r, n-r[n]$
 (c) $p, r, n-i$
22. Una de estas tres situaciones no es posible:
- (a) $f(n) \in O(n)$ y $f(n) \in \Omega(1)$
 (b) $f(n) \in \Omega(n^2)$ y $f(n) \in O(n)$
(c) $f(n) \in O(n)$ y $f(n) \in O(n^2)$
23. 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 falsa.
- (a) La complejidad temporal de la mejor solución posible al problema es $O(n!)$.
(b) 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.
 (c) La complejidad temporal de la mejor solución posible al problema es $O(n^2)$.

24. Cuál de los siguientes algoritmos proveería una cota pesimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).

- (a) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
- (b) Para todas las ciudades que son alcanzables en un paso desde la ciudad inicial, sumar la distancia a dicha ciudad y la distancia geométrica hasta la ciudad destino.
- (c) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.

25. Si para resolver un mismo problema usamos un algoritmo de vuelta atrás y lo modificamos mínimamente para convertirlo en un algoritmo de ramificación y poda, ¿qué cambiamos realmente?

- (a) Cambiamos la función que damos a la cota pesimista.
- (b) El algoritmo puede aprovechar mejor las cotas optimistas.
- (c) La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles.

26. Dadas las siguientes funciones:

```
// Precondición: { 0 <= i < v.size(); i < j <= v.size() }
unsigned f( const vector<unsigned>&v, unsigned i, unsigned j ) {
    if( i == j+1 )
        return v[i];
    unsigned sum = 0;
    for( unsigned k = 0; k < j - i; k++ )
        sum += f( v, i, i+k+1 ) + f( v, i+k+1, j );
    return sum;
}

unsigned g( const vector<unsigned>&v ) {
    return f( v, v.begin(), v.end() );
}
```

Se quiere reducir la complejidad temporal de la función g usando programación dinámica iterativa. ¿cuál sería la complejidad espacial?

- (a) cúbica
- (b) cuadrática
- (c) exponencial

27. En una cuadrícula se quiere dibujar el contorno de un cuadrado de n casillas de lado. ¿cuál será la complejidad temporal del mejor algoritmo que pueda existir?

- (a) $O(n^2)$
- (b) $O(n)$
- (c) $O(\sqrt{n})$

28. En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)
- (a) No, nunca es así.
 (b) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
(c) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
29. 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?
- (a) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
(b) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
 (c) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
30. Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solución de complejidad temporal polinómica a cualquier problema?
- (a) Sí, en cualquier caso.
 (b) No
(c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
31. En el esquema de *vuelta atrás* el orden en el que se van asignando los distintos valores a las componentes del vector que contendrá la solución...
- (a) ... es irrelevante si no se utilizan mecanismos de poda basados en la mejor solución hasta el momento.
(b) ... puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas.
 (c) Las otras dos opciones son ciertas.
32. La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la primera posición ...
- (a) ... no presenta caso mejor y peor para instancias del mismo tamaño.
(b) ... se comporta mejor cuando el vector ya está ordenado.
 (c) ... se comporta peor cuando el vector ya está ordenado.

33. ¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz (*greedy*) que es óptima?
- (a) El problema de la mochila discreta.
(b) El problema de la mochila continua o con fraccionamiento.
(c) El árbol de cobertura de coste mínimo de un grafo conexo.
34. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,
- (a) una estrategia voraz puede ser la única alternativa.
(b) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
(c) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
35. Dado un problema de optimización, el método voraz...
- (a) ... siempre obtiene la solución óptima.
(b) ... garantiza la solución óptima sólo para determinados problemas.
(c) ... siempre obtiene una solución factible.
36. La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...
- (a) ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
(b) El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.
(c) ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.
37. Se quieren 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*?
- (a) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$
(b) Sólo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
(c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

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

- (a) El problema de la asignación de tareas.
- (b) El problema del cambio.
- (c) La mochila discreta sin restricciones adicionales.

39. Di cuál de estos tres algoritmos no es un algoritmo de “divide y vencerás”

- (a) Quicksort
- (b) Mergesort
- (c) El algoritmo de Prim

40. Si $f(n) \in O(n^3)$, ¿puede pasar que $f(n) \in O(n^2)$?

- (a) No, porque $n^3 \notin O(n^2)$
- (b) Sólo para valores bajos de n
- (c) Es perfectamente posible, ya que $O(n^2) \subset O(n^3)$

1. Tratándose de un esquema general para resolver problemas de minimización, ¿qué falta en el hueco?:

```
Solution BB( Problem p ) {  
    Node best, init = initialNode(p);  
    Value pb = init.pessimistic_b();  
    priority_queue<Node>q.push(init);  
    while( ! q.empty() ) {  
        Node n = q.top(); q.pop();  
        q.pop();  
        if( ?????????? ){  
            pb = max( pb, n.pessimistic_b());  
            if( n.isTerminal() )  
                best = n.sol();  
            else  
                for( Node n : n.expand() )  
                    if( n.isFeasible())  
                        q.push(n);  
        }  
    return best;  
}
```

- (a) $n.optimistic_b() \geq pb$
- (b) $n.optimistic_b() \leq pb$
- (c) $n.pessimistic_b() \leq pb$

3. ¿Cuál de estas estrategias para calcular el n -ésimo elemento de la serie de Fibonacci ($f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$) es más eficiente?

- (a) Programación dinámica
- (b) La estrategia voraz
- (c) Para este problema, las dos estrategias citadas serían similares en cuanto a eficiencia

4. En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)
- (a) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
 - (b) Si, siempre es así.
 - (c) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
5. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...
- (a) ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.
 - (b) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
 - (c) ... determina la complejidad temporal en el peor de los casos del algoritmo.
15. El algoritmo de ordenación *Quicksort* divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?
- (a) $O(n \log n)$
 - (b) $\Omega(n)$ y $O(n^2)$
 - (c) $O(n)$
7. Uno de estos tres problemas no tiene una solución trivial y eficiente que siga el esquema voraz.
- (a) El problema del cambio.
 - (b) El problema de la mochila discreta sin limitación en la carga máxima de la mochila.
 - (c) El problema de la mochila continua.
8. Estudiad la relación de recurrencia:
- $$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{q}\right) + g(n) & \text{en otro caso} \end{cases}$$
- (donde p y q son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.
- (a) Divide y vencerás
 - (b) Ramificación y poda
 - (c) Programación dinámica

9. El siguiente programa resuelve el problema de cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n de manera que se maximice el precio de acuerdo con una tabla que da el precio para cada longitud, pero falta un trozo. ¿Qué debería ir en lugar de XXXXXXXX?

```

void fill(price m[]) {
    for (index i=0; i<=n; i++) m[i]=-1;

}

price cutrod(length n, price m[], price p[]) {
    price q;
    if (m[n]>=0) return m[n];
    if (n==0) q=0;
    else {
        q=-1;
        for (index i=1; i<=n; i++)
            q=max(q, p[i]+cutrod(XXXXXXXX));
    }
    m[n]=q;
    return q;
}

```

- (a) $n-m[n], m, p$
- (b)** $n-i, m, p$
- (c) $n, m[n]-1, p$

12. Sea $g(n) = \sum_{i=0}^K a_i n^i$. Di cuál de las siguientes afirmaciones es falsa:

- (a)** Las otras dos afirmaciones son ambas falsas.
- (b) $g(n) \in \Theta(n^K)$
- (c) $g(n) \in \Omega(n^K)$

13. Si $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$ entonces ...

- (a) $\dots f(n) \in \Theta(g(n))$
- (b)** $\dots f(n) \in O(g(n))$
- (c) $\dots g(n) \in O(f(n))$

14. ¿Cuál es la diferencia principal entre una solución de vuelta atrás y una solución de ramificación y poda para el problema de la mochila?

- (a) El hecho que la solución de ramificación y poda puede empezar con una solución subóptima voraz y la de vuelta atrás no.
- (b) El coste asintótico en el caso peor.
- (c)** El orden de exploración de las soluciones.

17. En un algoritmo de *ramificación y poda*, si la lista de nodos vivos no está ordenada de forma apropiada ...

- (a) ... podría ocurrir que se exploren nodos de forma innecesaria.
- (b) ... podría ocurrir que se descarten nodos factibles.
- (c) ... podría ocurrir que se pade el nodo que conduce a la solución óptima.

18. Sea la siguiente relación de recurrencia

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

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

- (a) $g(n) = n^2$
- (b) $g(n) = n$
- (c) $g(n) = 1$

19. Los algoritmos de *vuelta atrás* que hacen uso de cotas optimistas generan las soluciones posibles al problema mediante ...

- (a) ... 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.
- (b) ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.
- (c) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.

20. ¿Qué tienen en común el algoritmo que obtiene el k -ésimo elemento más pequeño de un vector (estudiado en clase) y el algoritmo de ordenación *Quicksort*?

- (a) El número de llamadas recursivas que se hacen.
- (b) La combinación de las soluciones a los subproblemas.
- (c) La división del problema en subproblemas.

21. ¿Cuál es el coste temporal asintótico de la siguiente función?

```
void f(int n, int arr[]) {  
    int i = 0, j = 0;  
    for(; i < n; ++i)  
        while(j < n && arr[i] < arr[j])  
            j++;  
}
```

- (a) $O(n^2)$
- (b) $O(n)$
- (c) $O(n \log n)$

22. ¿Se puede reducir el coste temporal de un algoritmo recursivo almacenando los resultados devueltos por las llamadas recursivas?

- (a) Sí, si se repiten llamadas a la función con los mismos argumentos
- (b) No, ello no reduce el coste temporal ya que las llamadas recursivas se deben realizar de cualquier manera
- (c) No, sólo se puede reducir el coste convirtiendo el algoritmo recursivo en iterativo

40. Sea n el número de elementos que contienen los vectores w y v en la siguiente función f . ¿Cuál es su complejidad temporal asintótica en función de n asumiendo que en la llamada inicial el parámetro i toma valor n ?

```
float f(vector <float>&w, vector<unsigned>&v,  
unsigned P, int i){  
    float S1, S2;  
    if (i>=0){  
        if (w[i] <= P)  
            S1= v[i] + f(w,v,P-w[i],i-1);  
        else S1= 0;  
        S2= f(w,v,P,i-1);  
        return max(S1,S2);  
    }  
    return 0;  
}
```

- (a) $\Theta(2^n)$
- (b) $\Omega(n)$ y $O(n^2)$
- (c) $\Omega(n)$ y $O(2^n)$

36. La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central ...

- (a) ... se comporta mejor cuando el vector ya está ordenado.
- (b) ...no presenta caso mejor y peor para instancias del mismo tamaño.
- (c) ... se comporta peor cuando el vector ya está ordenado.

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

- (a) Utilizar la solución (subóptima) que se obtiene al resolver el problema mediante un algoritmo voraz.
- (b) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
- (c) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.

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

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

Dí cuál de las siguientes afirmaciones es falsa:

- (a) Si $g(n) \in O(n^2)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda por inserción.
- (b) Si $g(n) \in O(n)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación *mergesort*.
- (c) Si $g(n) \in O(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

33. ¿Cuál es la definición correcta de $O(f)$?

- (a) $O(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ | \forall c \in \mathbb{R}, \forall n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \leq cg(n)\}$
- (b) $O(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq cf(n)\}$
- (c) $O(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \leq cg(n)\}$

34. Si $f(n) \in O(n^2)$, ¿podemos decir siempre que $f(n) \in O(n^3)$?

- (a) No, ya que $n^2 \notin O(n^3)$
- (b) Sí ya que $n^2 \in O(n^3)$
- (c) Sólo para valores bajos de n

37. Se desea encontrar el camino mas corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades como cota para limitar la búsqueda en un algoritmo de vuelta atrás. ¿Qué tipo de cota sería?

- (a) Una cota pesimista.
- (b) No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.
- (c) Una cota optimista.

39. 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 falsa.

- (a) 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.
- (b) La complejidad temporal de la mejor solución posible al problema es $O(n \log n)$.
- (c) La complejidad temporal de la mejor solución posible al problema está en $\Omega(n^2)$.

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

- (a) ... cuadrática en el caso peor.
- (b) ... exponencial en cualquier caso.
- (c) ... exponencial en el caso peor..

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

- (a) 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.
- (b) No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por tanto la eficiencia del algoritmo.
- (c) Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.

11. ¿Qué se entiende por *tamaño del problema*?

- (a) El valor máximo que puede tomar una instancia cualquiera de ese problema.
- (b) El número de parámetros que componen el problema.
- (c) La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.

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

- (a) $f(n) \in O(n^3)$
- (b) $f(n) \in \Theta(n^3)$
- (c) $f(n) \in \Theta(n^2)$

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

- (a) $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
- (b) $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$
- (c) $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$

14. ¿Cuál es la definición correcta de $\Omega(f)$?

- (a) $\Omega(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
- (b) $\Omega(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$
- (c) $\Omega(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$

15. Tratándose de un esquema general para resolver problemas de maximización, ¿qué falta en el hueco?:

```
Solution BB( Problem p ) {
    Node best, init = initialNode(p);
    Value pb = init.pessimistic_b();
    priority_queue<Node>q;
    q.push(init);
    while( ! q.empty() ) {
        Node n = q.top(); q.pop();
        q.pop();
        if( ?????????? ){
            pb = max( pb, n.pessimistic_b());
            if( n.isTerminal() )
                best = n.sol();
            else
                for( Node n : n.expand() )
                    if( n.isFeasible())
                        q.push(n);
        }
    }
    return best;
}
```

- (a) $n.\text{optimistic_b}() \leq pb$
(b) $n.\text{pessimistic_b}() \leq pb$
 (c) $n.\text{optimistic_b}() \geq pb$

16. 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) $\Theta(\log(n^2)) = \Theta(\log(n^3))$
(b) $\Theta(\log_2(n)) = \Theta(\log_3(n))$
 (c) $\Theta(\log^2(n)) = \Theta(\log^3(n))$

17. La función γ de un número semientero positivo (un número es semientero si al restarle 0.5 es entero) se define como:

```
double gamma( double n ) { // Se asume n>=0.5 y n-0.5 entero
    if( n == 0.5 )
        return sqrt(PI);
    return n * gamma( n - 1 );
}
```

¿Se puede calcular usando programación dinámica iterativa?

- (a) No, ya que el índice del almacén sería un número real y no entero.
(b) No, ya que no podríamos almacenar los resultados intermedios en el almacén.
 (c) Sí, pero la complejidad temporal no mejora.

21. ¿Cuál de estas afirmaciones es *falsa*?

- (a) La solución de programación dinámica iterativa al problema de la mochila discreta realiza cálculos innecesarios.
- (b) Los algoritmos iterativos de programación dinámica utilizan memoización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar.
- (c) La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.

22. El algoritmo de ordenación *Mergesort* divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?

- (a) $O(n)$
- (b) $O(n \log n)$
- (c) $O(1)$

23. 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 del nuevo algoritmo?

- (a) $n \log(n)$
- (b) $n \log^2(n)$
- (c) $n^2 \log(n)$

24. Los algoritmos voraces...

- (a) ... se basan en el hecho de que una organización apropiada de los datos permite descartar la mitad de ellos en cada paso.
- (b) ... se basan en el hecho de que se puede ahorrar esfuerzo guardando los resultados de cálculos anteriores en una tabla.
- (c) ... se basan en la idea de que la solución óptima se puede construir añadiendo repetidamente el mejor elemento disponible.

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

26. ¿Cuál de estas afirmaciones es *falsa*?

- (a) Hay problemas de optimización en los cuales el método voraz sólo obtiene la solución óptima para algunas instancias y un subóptimo para muchas otras instancias.
 - (b) Todos los problemas de optimización tienen una solución voraz que es óptima sea cual sea la instancia a resolver.
 - (c) Hay problemas de optimización para los cuales se puede obtener siempre la solución óptima utilizando una estrategia voraz.
31. 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) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
- (b) $(4^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$
- (c) $O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(n!)$

33. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ entonces

- (a) $f^2 \in \Theta(g_1 \cdot g_2)$
- (b) Las otras dos opciones son ambas ciertas.
- (c) $f \in \Theta(\max(g_1, g_2))$

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

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

Di cuál de las siguientes afirmaciones es *falsa*:

- (a) Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
 - (b) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación *quicksort*.
 - (c) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación *mergesort*.
35. ¿Para qué puede servir la cota pesimista de un nodo de *ramificación y poda*?
- (a) Para descartar el nodo si no es prometedor.
 - (b) Para obtener una cota optimista más precisa.
 - (c) Para actualizar el valor de la mejor solución hasta el momento.

39. Dada la relación de recurrencia:

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

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

- (a) $p > a^k$
- (b) $p = a^k$
- (c) $p < a^k$

1. 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 está en $\Omega(n^n)$.
 - (b) La complejidad temporal de la mejor solución posible al problema es $O(n \log n)$.
 - (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.
7. ¿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) $\Omega(n)$
- (b) $\Omega(1)$
- (c) Esta función no tiene caso mejor.

10. 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 ...

- (a) ... se van a utilizar tantos colores distintos a los ya utilizados como vértices quedan por colorear.
 (b) ... no se van a utilizar colores distintos a los ya utilizados.
(c) ... sólo va a ser necesario un color más.

12. 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) $\log(n^3) \notin \Theta(\log_3(n))$
(b) $\Theta(\log^2(n)) = \Theta(\log^3(n))$
 (c) $\Theta(\log_2(n)) = \Theta(\log_3(n))$

13. 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) Cuadrática
(b) Se puede reducir hasta lineal.
(c) La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.

14. 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) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
 (b) $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$
(c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

15. Dada la relación de recurrencia:

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

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

- (a) $p > a^k$
 (b) $p < a^k$
(c) $p = a^k$

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

- (a) $f(n) \in \Omega(n^3)$
 (b) $f(n) \in \Theta(n^2)$
(c) $f(n) \in \Theta(n^3)$

24. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ entonces

- (a) $f \in \Theta(g_1 \cdot g_2)$
(b) $f \notin \Theta(\max(g_1, g_2))$
 (c) $f \in \Theta(g_1 + g_2)$

32. 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) $\Theta(n)$
(b) $\Theta(\log_3 n)$
(c) Ninguna de las otras dos opciones es cierta.

25. ¿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) $\Theta(\log^2 n)$
(b) $\Theta(\log n)$
(c) $\Theta(n)$

26. 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) $\Theta(f) = O(f) \cap \Omega(f)$
(b) $\Omega(f) = \Theta(f) \cap O(f)$
(c) $O(f) = \Omega(f) \cap \Theta(f)$

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

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

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 mejor del algoritmo de ordenación *quicksort*.
- (b) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo 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 ordenación *mergesort*.
35. ¿Cuál de estas afirmaciones es *cierta*?
- (a) La ventaja de la solución de programación dinámica iterativa al problema de la mochila discreta es que nunca se realizan cálculos innecesarios.
- (b) La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.
- (c) Los algoritmos iterativos de programación dinámica utilizan memoización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar.

1. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?

- (a) No, con este esquema se puede conocer el número total de caminos distintos que conducen a la salida pero no se puede saber la composición de ninguno de ellos.
- (b) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
- (c) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
2. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
- (a) ... debe recorrer siempre todo el árbol.
- (b) ... no se puede usar para resolver problemas de optimización.
- (c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

3. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

- (a) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- (b) El valor de la mochila continua correspondiente .
- (c) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

4. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continúe ordenado es...

- (a) ... $\Omega(n^2)$.
- (b) ... $O(n)$.
- (c) ... $O(\log n)$.

5. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?

- (a) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
- (b) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.
- (c) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.

6. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?

- (a) Explorar primero los nodos que están más completados.
- (b) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
- (c) Explorar primero los nodos con mejor cota optimista.

7. Sea la siguiente relación de recurrencia

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

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

- (a) Las otras dos opciones son ambas ciertas.
- (b) $g(n) = \log n$
- (c) $g(n) = \sqrt{n}$

8. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces

- (a) $f \in \Omega(g_1 + g_2)$
- (b) $f \notin \Omega(\min(g_1, g_2))$
- (c) $f \in \Omega(g_1 \cdot g_2)$

9. Dada la siguiente función:

```
int exa (string & cad, int pri, int ult){  
    if (pri>=ult)  
        return 1;  
    else  
        if (cad[pri]==cad[ult])  
            return exa(cad, pri+1, ult-1);  
        else  
            return 0;  
}
```

¿Cuál es su complejidad temporal asintótica?

- (a) $O(n^2)$
- (b) $O(n)$
- (c) $O(\log n)$

10. El esquema de vuelta atrás ...

- (a) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
- (b) Las otras dos opciones son ambas verdaderas.
- (c) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.

11. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?

- (a) Cola de prioridad
- (b) Pila
- (c) Cola

12. ¿Qué nos proporciona la media entre el coste temporal asintótico (o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?

- (a) Nada de interés.
- (b) El coste temporal promedio.
- (c) El coste temporal asintótico en el caso medio.

14. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?

- (a) $O(n^3 \log n)$
- (b) $O(n^3)$
- (c) $O(n \log n)$

13. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica un esquema de programación dinámica. En cuanto a la complejidad temporal, ¿cuál es la mejora de la versión recursiva con memoización frente a la recursiva ingenua que se obtiene a partir del esquema divide y vencerás?

- (a) La mejora no está garantizada puesto que la versión recursiva con memoización podría ser peor que la obtenida a partir del esquema divide y vencerás.
- (b) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
- (c) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.

15. Dada la siguiente función:

```
int exa (vector <int>& v){  
    int j, i=1, n=v.size();  
  
    if (n>1) do{  
        int x = v[i];  
        for (j=i; j >0 and v[j-1] >x; j--)  
            v[j]=v[j-1];  
        v[j]=x;  
        i++;  
    } while (i<n);  
    return 0;  
}
```

Marcad la opción correcta.

- (a) La complejidad temporal exacta es $\Theta(n^2)$
- (b) La complejidad temporal en el mejor de los casos es $\Omega(n)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

16. El esquema voraz ...

- (a) Las otras dos opciones son ambas falsas.
- (b) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
- (c) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.

17. Cuando la descomposición de un problema da lugar a subproblemas de tamaño similar al original, muchos de los cuales se repiten, ¿qué esquema es a priori más apropiado?

- (a) Ramificación y poda.
- (b) Divide y vencerás.
- (c) Programación dinámica.

18. Dada la siguiente función (donde $\max(a, b) \in \Theta(1)$):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
    float a, b;
    if (i>=0){
        if (p[i] <= P)
            a= v[i]+exa(v,p,P-p[i],i-1);
        else a= 0;
        b= exa(v,p,P,i-1);
        return max(a,b);
    }
    return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el peor de los casos es $O(n^2)$
- (b) La complejidad temporal en el peor de los casos es $O(2^n)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(n^2)$

19. ¿Cuál sería la complejidad temporal de la siguiente función tras aplicar programación dinámica?

```
double f(int n, int m) {
    if(n == 0) return 1;
    return m * f(n-1,m) * f(n-2,m);
}
```

- (a) $\Theta(n)$
- (b) $\Theta(n^2)$
- (c) $\Theta(n \times m)$

20. Dado el problema del laberinto con tres movimientos, se pretende conocer la longitud del camino de salida más corto. Para ello se aplica el esquema voraz con un criterio de selección que consiste en elegir primero el movimiento *Este* siempre que la casilla sea accesible. Si no lo es se descarta ese movimiento y se prueba con *Sureste* y por último, si este tampoco es posible, se escoge el movimiento *Sur*. ¿Qué se puede decir del algoritmo obtenido?

- (a) Que es un algoritmo voraz pero sin garantía de solucionar el problema.
- (b) Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.
- (c) Que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.

21. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- (a) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$
- (b) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$
- (c) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$

22. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?

- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
- (b) Programación dinámica.
- (c) Divide y vencerás.

23. De las siguientes afirmaciones marca la que es verdadera.

- (a) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
- (b) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.
- (c) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.

24. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?

- (a) Asegura un ahorro en la comprobación de todas las soluciones factibles.
- (b) Siempre es mayor o igual que la mejor solución posible alcanzada.
- (c) Las otras dos opciones son ambas falsas.

25. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?

- (a) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
- (b) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
- (c) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$

26. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?

- (a) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
- (b) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
- (c) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$

27. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
- (a) Vuelta atrás, es el esquema más eficiente para este problema.
 - (b) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
 - (c) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
28. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica un esquema de divide y vencerás. ¿Cuál sería la recurrencia apropiada para el caso general?
- (a) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$
 - (b) $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$
 - (c) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
29. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
 - (b) un vector de booleanos.
 - (c) un par de enteros que indiquen los cortes realizados y el valor acumulado.
30. 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) $\Theta(n) \subset \Theta(n^2)$
 - (b) $n + n \log n \in \Omega(n)$
 - (c) $O(2^{\log n}) \subset O(n^2)$
31. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) Las otras dos opciones son ambas verdaderas.
 - (b) ... pueden eliminar vectores que representan posibles soluciones factibles.
 - (c) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.

32. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
 - (b) Lo más adecuado sería usar una técnica de ramificación y poda ya que es muy importante el orden en el que se exploran las soluciones parciales.
 - (c) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
33. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
- (a) Que se podría explorar menos nodos de los necesarios.
 - (b) Que se podría podar el nodo que conduce a la solución óptima.
 - (c) Que se podría explorar más nodos de los necesarios.
34. Un algoritmo recursivo basado en el esquema *divide y vencerás...*
- (a) Las otras dos opciones son ambas verdaderas.
 - (b) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - (c) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
35. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recurrente expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?
- (a) $T(n) = 2T(n - 1) + 1$
 - (b) $T(n) = 2T(n - 1) + n$
 - (c) $T(n) = T(n - 1) + n$
36. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?
- (a) $n \log^2 n$
 - (b) n^2
 - (c) $n \log n$

37. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?

- (a) $\Theta(n \log n)$
- (b) $\Theta(n^2)$
- (c) Ninguna de las otras dos opciones es cierta.

38. Dada la siguiente función:

```
int exa (vector <int>& v) {
    int i,sum=0, n=v.size();

    if (n>0){
        int j=n;
        while (sum<100){
            j=j/2;
            sum=0;
            for (i=j;i<n;i++)
                sum+=v[i];
            if (j==0) sum=100;
        }
        return j;
    }
    else return -1;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal exacta es $\Theta(n \log n)$
- (b) La complejidad temporal en el mejor de los casos es $\Omega(1)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(n)$

39. Dado el problema del laberinto con tres movimientos, ¿cuál de las estrategias siguientes proveería de una cota optimista para ramificación y poda?

- (a) Suponer que ya no se van a realizar más movimientos.
- (b) Las otras dos estrategias son ambas válidas.
- (c) Suponer que en adelante todas las casillas del laberinto son accesibles.

40. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

- (a) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
- (b) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
- (c) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

1. Los algoritmos de ordenación *Quicksort* y *Mergesort* tienen en común...
 - (a) ... que ordenan el vector sin usar espacio adicional.
 - (b) ... que se ejecutan en tiempo $O(n)$.
 - (c) ... que aplican la estrategia de *divide y vencerás*.
3. Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima.
 - (a) Lo más adecuado sería usar una técnica de ramificación y poda, aunque en el peor caso el coste temporal asintótico (o complejidad temporal) sería exponencial.
 - (b) Para encontrar la solución habría que probar con todas las combinaciones posibles de m enteros, con lo que la técnica de ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Una técnica voraz daría una solución óptima.
4. Di cuál de estos resultados de coste temporal asintótico es falsa:
 - (a) La ordenación de un vector usando el algoritmo *Quicksort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - (b) La ordenación de un vector usando el algoritmo *Mergesort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - (c) La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$.
6. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
 - (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - (c) Las dos anteriores son verdaderas.
7. Tenemos n substancias diferentes en polvo y queremos generar todas las distintas formas de mezclarlas de forma que el peso total no supere un gramo. Como la balanza que tenemos solo tiene una precisión de 0.1 gramos, no se considerarán pesos que no sean múltiplos de esta cantidad. Queremos hacer un programa que genere todas las combinaciones posibles.

[A] o [C]

 - (a) No hay ningún problema en usar una técnica de vuelta atrás.
 - (b) No se puede usar vuelta atrás porque las decisiones no son valores discretos.
 - (c) No se puede usar vuelta atrás porque el número de combinaciones es infinito.

9. Dado un problema de optimización, ¿cuándo se puede aplicar el método de vuelta atrás?

- (a) Es condición necesaria (aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable.
- (b) Es condición necesaria y suficiente que el dominio de las decisiones sea discreto o discretizable.
- (c) No sólo es condición necesaria que el dominio de las decisiones sea discreto o discretizable; además, debe cumplirse que se puedan emplear mecanismos de poda basados en la mejor solución hasta el momento.

10. ¿Cuál de estas tres expresiones es cierta?

- (a) $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
- (b) $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- (c) $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$

11. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

- (a) $f(n) \in \Theta(n^2)$
- (b) $f(n) \in \Theta(n \log n)$
- (c) $f(n) \in \Theta(n)$

12. Indicad cuál de estas tres expresiones es falsa:

- (a) $\Theta(n/2) = \Theta(n)$
- (b) $\Theta(n) \subset O(n)$
- (c) $\Theta(n) \subset \Theta(n^2)$

13. Indica cuál es el coste temporal asintótico (o complejidad temporal), en función de n , del programa siguiente:

```
s=0; for(i=0; i<n; i++) for(j=i; j<n; j++) s+=n*i*j;
```

- (a) Es $O(n^2)$ pero no $\Omega(n^2)$.
- (b) Es $\Theta(n^2)$
- (c) Es $\Theta(n)$

14. Un programa con dos bucles anidados uno dentro del otro, cada uno de los cuales hace aproximadamente n iteraciones, tarda un tiempo

- (a) $O(n^2)$
- (b) $O(2^n)$
- (c) $O(n)$

15. La eficiencia de los algoritmos voraces se basa en...

- (a) ... el hecho de que, con antelación, las posibles decisiones se ordenan de mejor a peor.
- (b) ... el hecho de que las decisiones tomadas no se reconsideran.
- (c) En el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema.

17. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n - 1) + 1$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

- (a) $f(n) \in \Theta(n^2)$
- (b) $f(n) \in \Theta(2^n)$
- (c) $f(n) \in \Theta(n)$

18. Pertenece $3n^2 + 3$ a $O(n^3)$?

- (a) No.
- (b) Sólo para $c = 1$ y $n_0 = 5$
- (c) Sí.

19. Las relaciones de recurrencia...

- (a) ... aparecen sólo cuando la solución sea del tipo *divide y vencerás*.
- (b) ... expresan recursivamente el coste temporal de un algoritmo.
- (c) ... sirven para reducir el coste temporal de una solución cuando es prohibitivo.

20. El coste temporal de un algoritmo se ajusta a la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 0 \\ n + \sum_{j=0}^{n-1} T(j) & n > 1 \end{cases}$$

¿qué coste temporal asintótico (o complejidad temporal) tendrá el algoritmo?

- (a) $O(n \log(n))$
- (b) $O(n^2)$
- (c) $O(2^n)$

22. ¿Cuál es el coste espacial asintótico del siguiente algoritmo?

```
int f( int n ) {
    int a = 1, r = 0;
    for( int i = 0; i < n; i++ ) {
        r = a + r;
        a = 2 * r;
    }
    return r;
}
```

- (a) $O(1)$
- (b) $O(\log(n))$ [A] o [C]
- (c) $O(n)$

23. De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

NO SEGURO

- (a) El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible
- (b) El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando las mínimas monedas.
- (c) El problema del viajante de comercio

24. ¿Qué algoritmo es asintóticamente más rápido, el *quicksort* o el *mergesort*?

- (a) como su nombre indica, el *quicksort*.
- (b) son los dos igual de rápidos, ya que el coste temporal asintótico de ambos es $O(n \log(n))$.
- (c) el *mergesort* es siempre más rápido o igual (salvo una constante) que el *quicksort*.

25. El coste temporal del algoritmo de ordenación por inserción es...

- (a) ... $O(n^2)$.
- (b) ... $O(n)$.
- (c) ... $O(n \log n)$.

26. Los algoritmos de programación dinámica hacen uso...

- (a) ...de que la solución óptima se puede construir añadiendo el componente óptimo de los restantes, uno a uno.
- (b) ...de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores.
- (c) ...de una estrategia trivial consistente en examinar todas las soluciones posibles.

27. ¿Cuál de estos tres problemas de optimización no tiene una solución voraz (*greedy*) que sea óptima?

- (a) El problema de la mochila continua o con fraccionamiento.
- (b) El problema de la mochila discreta.
- (c) El árbol de cobertura de coste mínimo de un grafo conexo.

28. El problema de la función compuesta mínima consiste en encontrar, a partir de un conjunto de funciones dadas, la secuencia mínima de composiciones de éstas que permita transformar un número n en otro m . Se quiere resolver mediante ramificación y poda. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- (a) Mediante un vector de booleanos.
- (b) Mediante un vector de reales.
- (c) Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.

29. ¿Cuál de estas tres expresiones es falsa?

- (a) $2n^2 + 3n + 1 \in O(n^3)$
- (b) $n + n \log(n) \in \Omega(n)$
- (c) $n + n \log(n) \in \Theta(n)$

30. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- (a) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$
- (b) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$
- (c) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$

36. La solución recursiva ingenua a un determinado problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en...

- (a) ... un algoritmo del estilo de *divide y vencerás*.
- (b) ... un algoritmo de programación dinámica.
- (c) ... un algoritmo voraz.

Un algoritmo recursivo basado en el esquema *divide y vencerás*...

Seleccione una:

- a ... será más eficiente cuanto más equitativa sea la división en subproblemas
- b. Las demás opciones son verdaderas.
- c. ... nunca tendrá una complejidad exponencial.

¿Cuál de estas tres expresiones es falsa?

Seleccione una

- a. $3n^2 + 1 \in O(n^3)$
- b. $n + n \log(n) \in \Omega(n)$
- c. $n + n \log(n) \in \Theta(n)$ ✓

Indica cuál es la complejidad en el peor caso de la función replace:

```
unsigned bound( const vector<int> &v ) {
    for( unsigned i = 0; i < v.size(); i++ )
        if( v[i] == '0' )
            return i;
    return v.size();
}

void replace( vector<int>& v, int c ) {
    for( unsigned i = 0; i < bound(v); i++ )
        v[i] = c;
}
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$ ✓
- c. $O(n)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
    if (n<=1)
        return 0;
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);
    for (unsigned i=1; i<n-1; i++)
        for (unsigned j=1; j<=i; j++)
            sum+=i*j;
    return sum;
}
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $\Theta(2^n)$
- c. $\Theta(n^2 \log n)$

Será $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + 1$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^2)$
- c. $f(n) \in \Theta(n \log(n))$

Considerad estos dos fragmentos:

```
s=0;for (i=0;i<n;i++) s+=i;
```

y

```
s=0;for (i=0;i<n;i++) if (a[i] != 0) s+=i;
```

y un array $a[i]$ de números enteros. Indicad cuál de estas tres afirmaciones es cierta:

Seleccione una:

- a. El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.
- b. El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.
- c. El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero.

Indica cuál es la complejidad, en función de γ_2 , del fragmento siguiente:

```
int a = 0;
for( int i = 0; i < n; i++ )
    for( int j = i; j > 0; j /= 2 )
        a += A[i][j];
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n)$
- c. $O(n^2)$

Indica cuál es la complejidad en función de γ_2 , donde K es una constante (no depende de γ_2), del fragmento siguiente :

```
for( int i = k; i < n - k; i++){
    A[i] = 0;
    for( int j = i - k; j < i + k; j++ )
        A[i] += B[j];
}
```

Seleccione una:

- a. $O(n)$
- b. $O(n \log n)$
- c. $O(n^2)$

La versión de Quicksort que utiliza como pivote la mediana del vector ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ \sqrt{n} + 3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(\sqrt{n} \log n)$

Un programa con dos bucles anidados uno dentro del otro. El primero hace γ_3 iteraciones aproximadamente y el segundo la mitad, tarda un tiempo

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$
- c. $O(n \sqrt{n})$

Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en nuevo de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2 \log n)$
- c. $O(n^2)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){  
    if (n<=1)  
        return 0;  
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);  
    for (unsigned i=1; i<n-1; i++)  
        for (unsigned j=i; j<-i; j++)  
            for (unsigned k=1; k<=j; k++)  
                sum+=i*j*k;  
    return sum;  
}
```

Seleccione una:

- a. $\Theta(2^n)$
- b. $\Theta(n^3 \log n)$
- c. $\Theta(n^3)$ ✓

Indica cuál es la complejidad de la función siguiente:

```
unsigned sum( const mat &A ) { // A es una matriz cuadrada  
    unsigned d = A.n_rows();  
    unsigned a = 0;  
    for( unsigned i = 0; i < d; i++ )  
        for( unsigned j = 0; j < d; j++ )  
            a += A(i,j);  
    return a;  
}
```

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$ ✓
- c. $O(n \log n)$

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno
- b. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén ✓
- c. ... de una estrategia trivial consistente en examinar todas las soluciones posibles

Cuando se calculan los coeficientes binomiales usando la recursión $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$, con $\binom{n}{0} = \binom{n}{n} = 1$, qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- c. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$: $f(1) = 1$. Indicad cuál de estas

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n)$
- c. $f(n) \in \Theta(n \log(n))$ ✓

Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...

Seleccione una:

- a. ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
- b. ... es condición necesaria que existan instancias distintas del problema con el mismo tamaño. ✓
- c. ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

Un problema de tamaño n puede transformarse en tiempo $O(n)$ en siete de tamaño $n/7$; por otro lado, la solución al problema cuando la talle es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$
- c. $O(n \log n)$ ✓

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. Las demás opciones son falsas. ✓
- c. ... siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursivo.

Considerad la función siguiente:

```
int M( int i, int f) {
    if ( i == f )
        return i;
    else {
        e = v[ M( i, (i+f)/2 ) ];
        f = v[ M( (i+f)/2+1, f ) ];
        if (e<f)
            return e;
        else
            return f;
    }
}
```

Si la talla del problema viene dada por $n = f - i + 1$, ¿cuál es el coste temporal asintótico en el supuesto de que n sea una potencia de 2?

Seleccione una:

- a. $O(n)$ ✓
- b. $O(n^2)$.
- c. $O(n \log(n))$.

El coste temporal asintótico del fragmento

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

y el del fragmento

```
s=0; for(i=0;i<n;i++) for(j=0;j<n;j++) s+=i*i*j;
```

son ...

Seleccione una:

- a. ... iguales. ✓
- b. ... el del segundo, menor que el del primero.
- c. ... el del primero, menor que el del segundo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n^2 + 3f(n/3) & n > 1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n^2 \log n)$ ✗
- c. $f(n) \in \Theta(n)$

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado. ✓
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la posición central ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado. ✓
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... no presenta casos mejor y peor distintos para instancias del mismo tamaño.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n+3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n \log n)$ ✓
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(n)$

Cuando se resuelve usando un algoritmo de *vuelta atrás* un problema de γ_2 decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a. $O(2^n)$ ✓
- b. $O(n!)$
- c. $O(n^2)$

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de γ_2 vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se multiplica γ_2 por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las γ_2 aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar.
- c. ... es exponencial con el número de decisiones a tomar. ✓

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... 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.
- b. Las otras dos opciones son verdaderas. ✓
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

Cuando resolvemos un problema mediante un esquema de *ramificación y poda* ...

Seleccione una:

- a. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito. ✓
- b. ... las decisiones sólo pueden ser binarias.
- c. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b. ... sólo si se usa para resolver problemas de optimización. ✓
- c. ... para determinar si una solución es factible.

En la estrategia de *ramificación y poda* ...

Seleccione una:

- a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. ✓
- b. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

Si para resolver un mismo problema usamos un algoritmo de *ramificación y poda* y lo modificamos mínimamente para convertirlo en un algoritmo de *vuelta atrás*, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Sería necesario comprobar si las soluciones son factibles o no puesto que *ramificación y poda* sólo genera nodos factibles.

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no se puede usar para resolver problemas de optimización.
- b. ... debe recorrer siempre todo el árbol.
- c. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

La estrategia de *vuelta atrás* es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b. El espacio de soluciones es un conjunto infinito.
- c. El espacio de soluciones es un conjunto finito. ✓

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c. El valor óptimo de la mochila continua correspondiente.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de la mochila continua correspondiente.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota **pesimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones
- c. Sería una cota **optimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible

Se desea encontrar el camino mas corto entre dos ciudades

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz, ¿sería como cota pesimista?

Seleccione una:

- a. No, ya que no asegura que se encuentre una solución factible. ✓
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible
- c. No, ya que en algunos casos puede dar distancias menores que la óptima

El problema de cortar un tubo de longitud n en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. ✓
- c. ... se debe resolver mediante un algoritmo de *vuelta atrás*. dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar $O(n!)$ posibilidades.

Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de n vértices donde cada arista tiene un coste asignado. X
- b. La solución de *ramificación y poda* al problema de la asignación de n tareas a n trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo. X
- c. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de n vértices de forma que el coste es mínimo (*minimum spanning tree*). ✓

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos

Seleccione una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz. ✓
- b. un algoritmo divide y vencerás.
- c. un algoritmo de programación dinámica.

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de $O(n)$ a $O(1)$. donde n es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de $O(n^2)$ a $O(n \log n)$, donde n es el número de objetos a considerar.

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a. $T(n) \in \Theta(n^2)$
- b. $T(n) \in \Theta(n!)$
- c. $T(n) \in \Theta(2^n)$ ✓

En el método voraz ...

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- c. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f( unsigned x, unsigned v[] ) {  
    if (x==0)  
        return 0;  
    unsigned m = 0;  
    for ( unsigned k = 0; k < x; k++ )  
        m = max( m, v[k] + f( x-k, v ) );  
    return m;  
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A
- b. int A[] ✓
- c. int A[][]

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f( unsigned y, unsigned x){ // suponemos y >= x  
    if (x==0 || y==x) return 1;  
    return f(y-1, x-1) + f(y-1, x);  
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[]
- b. int A
- c. int A[][] ✓

La solución de programación dinámica iterativa del problema de la mochila discreta

Seleccione una:

- a. calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva
- b. tiene un coste temporal asintótico exponencial con respecto al número de objetos
- c. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. ... se puede resolver siempre con una estrategia voraz. ✓

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... al menos una solución factible está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... la solución óptima está garantizada.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f( unsigned y, unsigned x){ // suponemos y >= x
    if (x==0 || y==x) return 1;
    return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a. $O(1)$
- b. $O(y^2)$
- c. $O(y)$ ✓

El valor que se obtiene con el método voraz para el problema de la mochila discreta es

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo $\Theta(2^n)$
- b. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud $j < n$ el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo $\Theta(n!)$. ✓

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f( int x, int y ) {
    if( x <= y ) return 1;
    return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a. $O(x^2)$
- b. $O(1)$ ✓
- c. $O(x)$

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y) {
    if( y < 0 ) return 0;
    float A = 0.0;
    if ( v1[y] <= x )
        A = v2[y] + f( x-v1[y], y-1 );
    float B = f( x, y-1 );
    return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a. $O(y^2)$
- b. $O(1)$
- c. $O(y)$ ✓

La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. Las otras dos opciones son ciertas. ✓
- c. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la mochila continua. ✓
- b. El fontanero diligente y la asignación de tareas
- c. El fontanero diligente y el problema del cambio.

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene una solución factible.
- b. ... siempre obtiene la solución óptima.
- c. ... garantiza la solución óptima sólo para determinados problemas. ✓

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. $O(1) \subset O(\lg n) \subset O(\lg \lg n) \subset \dots$ b. $\dots \subset (n!) \subset O(2^n) \subset O(n^n)$ c. $\dots \subset (2^n) \subset O(n!) \subset O(n^p)$	
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. $O(n\lg n)$ 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 qué 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 $f(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_0$, con $a_m \in \mathbb{R}^+$ 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. $f1(n) \cdot f2(n) \in O(\max(g1(n), g2(n)))$ 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. $f1(n) + f2(n) \in O(\max(g1(n), g2(n)))$ 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	$f(n) = 5n + 3m \cdot n + 11$ entonces $f(n)$ pertenece a: a. $O(n \cdot m)$. 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 $\geq a^2$: 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 $f(n) \in \Omega(g(n))$ entonces: a. $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n \geq n_0$ b. $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n$ c. $\exists c, n_0 \in \mathbb{R}^+: f(n) \leq c \cdot g(n) \forall n \geq n_0$	
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. $\Theta(n \lg n)$

DIVIDE Y VENCERAS Y PROGRAMACIÓN DINAMICA: EJERCICIOS

N	ENUNCIADO
1	¿Qué esquema algorítmico utiliza el algoritmo de ordenación Quicksort? a. Divide y Vencerás b. Programación Dinámica c. Backtracking
2	Ante un problema que presenta una solución recursiva siempre podemos aplicar: a. Divide y vencerás b. Programación dinámica c. Cualquiera de las dos anteriores
3	En cual de los siguientes casos no se puede aplicar el esquema Divide y Vencerás: a. Cuando los subproblemas son de tamaños muy diferentes b. Cuando el problema no cumple el principio de optimalidad c. Se puede aplicar en ambos casos.
4	Dado el algoritmo de búsqueda binaria, supongamos que, en vez de dividir la lista de elementos en dos mitades del mismo tamaño, la dividimos en dos partes de tamaños 1/3 y 2/3. El coste de este algoritmo: a. Es el mismo que el del original b. Es mayor que el del original c. Es menor que el del original
5	Si n es el número de elementos del vector, el coste del algoritmo Mergesort es: a. $O(n^2)$ y $\Omega(n \log n)$ b. $\Theta(n \log n)$ c. $\Theta(n^2)$
6	Un problema se puede resolver por Divide y Vencerás siempre que: a. Cumpla el principio de optimalidad b. Cumpla el teorema de reducción c. Ninguna de las anteriores
7	La serie de números de Fibonacci se define de la siguiente forma: $fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$ Para implementar esta función podemos emplear : a. Divide y vencerás b. Programación dinámica c. Cualquiera de las dos anteriores
8	La serie de números de Fibonacci se define de la siguiente forma: $fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$ ¿Qué implementación de entre las siguientes supone el menor coste? a. Divide y vencerás b. Programación dinámica c. Ambas tienen el mismo coste asintótico
9	El problema de la mochila, ¿puede solucionarse de forma óptima empleando la estrategia de divide y vencerás?: a. Sólo para el caso de la mochila con fraccionamiento b. Sólo para el caso de la mochila sin fraccionamiento c. Si, se puede aplicar para ambos casos.
10	Para que un problema de optimización se pueda resolver mediante PD es necesario que: a. Cumpla el principio de optimalidad b. Cumpla el teorema de reducción c. Cumpla los dos anteriores
11	Dada una solución recursiva a un problema ¿Cómo podemos evitar la resolución de los mismos subproblemas muchas veces? a. Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas pequeños. b. Resolver los subproblemas de menor a mayor y guardar su resultado en una tabla, inicializándola con los problemas pequeños. c. Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas más grandes.
12	Si aplicamos Programación Dinámica a un problema que también tiene solución por divide y vencerás podemos asegurar que... a. El coste temporal se reduce y el espacial aumenta con respecto a la solución por DyV b. El coste temporal aumenta y el espacial se reduce con respecto a la solución por DyV c. Ninguna de las anteriores.
13	¿Cuándo utilizaremos Programación Dinámica en lugar de Divide y Vencerás? a. Cuando se incrementa la eficacia b. Cuando se incrementa la eficiencia c. Cuando se reduce el coste espacial.
14	En programación dinámica, dónde almacenamos los valores de los problemas resueltos? a. En un vector unidimensional b. En un vector bidimensional c. Depende del problema

15	Supongamos el problema de la mochila resuelto mediante Programación Dinámica y particularizado para n elementos y un peso máximo transportable de P. ¿Es necesario calcular valores para toda la matriz auxiliar para obtener el resultado? <ol style="list-style-type: none"> Si No Depende de los valores de n y P.
16	Un problema de optimización cuya solución se puede expresar mediante una secuencia de decisiones cumple el principio de optimalidad si, dada una secuencia óptima: <ol style="list-style-type: none"> Existe una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado Existe al menos una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado Cualquier subsecuencia de esa solución corresponde a la solución óptima de su subproblema asociado
17	La programación dinámica, para resolver un problema, aplica la estrategia... <ol style="list-style-type: none"> Se resuelven los problemas más pequeños y, combinando las soluciones, se obtienen las soluciones de problemas sucesivamente más grandes hasta llegar al problema original. Se descompone el problema a resolver en subproblemas más pequeños, que se resuelven independientemente para finalmente combinar las soluciones de los subproblemas para obtener la solución del problema original. Ninguna de las anteriores
18	¿Qué esquema de programación es el adecuado para resolver el problema del k-ésimo mínimo en un vector? <ol style="list-style-type: none"> Programación Dinámica Divide y Vencerás Ninguno de los dos
19	Si n es el número de elementos de un vector. La solución de menor coste al problema de encontrar su k-ésimo mínimo tiene la siguiente complejidad: <ol style="list-style-type: none"> $\Omega(n)$ y $O(n \log n)$ $\Omega(n)$ y $O(n^2)$ Ninguna de las dos
20	Si n es el número de elementos de un vector. Podemos encontrar una solución al problema de encontrar su k-ésimo que esté acotada superiormente por : <ol style="list-style-type: none"> $O(n^3)$ $O(n)$ Ninguna de las dos
21	Dada la solución recursiva al problema de encontrar el k-ésimo mínimo de un vector. Cada llamada recursiva, ¿cuántas nuevas llamadas recursivas genera? <ol style="list-style-type: none"> una o ninguna dos o ninguna una o dos
22	La solución al problema de encontrar el k-ésimo mínimo de un vector pone en práctica la siguiente estrategia: <ol style="list-style-type: none"> Ordena totalmente el vector Ordena parcialmente el vector No ordena ningún elemento del vector
23	¿Qué esquema de programación es el adecuado para resolver el problema de la búsqueda binaria? <ol style="list-style-type: none"> Programación Dinámica Divide y Vencerás Ninguno de los dos
24	Si n es el número de elementos de un vector. La solución de menor coste al problema de la búsqueda binaria tiene la siguiente complejidad: <ol style="list-style-type: none"> $\Omega(\log n)$ y $O(n \log n)$ $\Theta(n \log n)$ $\Omega(1)$ y $O(\log n)$
25	¿Con qué esquema de programación obtenemos algoritmos que calculan la distancia de edición entre dos cadenas? <ol style="list-style-type: none"> Programación Dinámica Divide y vencerás Ambos
26	Disponemos de dos cadenas de longitudes m y n. Si resolvemos el problema de la distancia de edición mediante programación dinámica, ¿De qué tamaño debemos definir la matriz que necesitaremos? <ol style="list-style-type: none"> $(m-1) \times (n-1)$ $m \times n$ $(m+1) \times (n+1)$

ALGORITMOS VORACES, VUELTA ATRÁS Y RAMIFICACIÓN Y PODA: EJERCICIOS

N	ENUNCIADO
1	El método voraz se emplea en la resolución de problemas de selección y optimización en los que se pretende encontrar: <ul style="list-style-type: none"> a. Una solución que satisfaga unas restricciones y optimice una cierta función objetivo. b. Todas las soluciones que satisfagan unas restricciones. c. La dos respuestas anteriores son correctas.
2	Voraz siempre da solución óptima: <ul style="list-style-type: none"> a. Al problema de la mochila sin fraccionamiento. b. Al problema de la mochila con fraccionamiento. c. A los dos.
3	En el método Voraz, aunque las decisiones son irreversibles, podemos asegurar que: <ul style="list-style-type: none"> a. Siempre obtendremos la solución óptima. b. Siempre obtendremos una solución factible. c. Sólo obtendremos la solución óptima para algunos problemas.
4	Dado un problema de optimización y un algoritmo Voraz que lo soluciona, ¿cuándo podemos estar seguros de que la solución obtenida será óptima?: <ul style="list-style-type: none"> a. Cuando demostremos formalmente que el criterio conduce a una solución óptima para cualquier instancia del problema. b. Voraz siempre encuentra solución óptima. c. En ambos casos. Las dos son correctas
5	Si aplicamos un algoritmo voraz que no nos garantiza la solución óptima sobre un problema entonces... <ul style="list-style-type: none"> a. Obtendremos una solución factible. b. Puede que no encuentre ninguna solución aunque ésta exista. c. Si el problema tiene solución óptima, el esquema voraz nos garantiza que la encuentra.
6	El problema de la mochila, ¿encuentra su solución óptima empleando la estrategia voraz?: <ul style="list-style-type: none"> a. Sólo para el caso de la mochila con fraccionamiento b. Sólo para el caso de la mochila sin fraccionamiento c. En cualquiera de los casos anteriores.
7	Dado un grafo G que representa las poblaciones de la provincia de Alicante de más de 20.000 habitantes junto con todas las carreteras de conexión entre ellas. Queremos obtener el recorrido que nos permita pasar por todas estas ciudades una única vez y volver al punto de origen recorriendo el mínimo número de kilómetros. Si aplicamos una estrategia voraz sobre este grafo obtendremos... <ul style="list-style-type: none"> a. Una solución factible b. La solución óptima c. Puede que no encuentre ninguna solución aunque ésta exista.
8	Al aplicar backtracking obtenemos la solución óptima a un problema: <ul style="list-style-type: none"> a. Siempre b. En algunos casos c. Sólo cuando el problema cumple el principio de Optimalidad.
9	Si aplicamos un esquema backtracking que no nos garantiza la solución óptima sobre un problema entonces... <ul style="list-style-type: none"> a. Obtendremos una solución factible. b. Puede que no encuentre ninguna solución aunque ésta exista. c. Ninguna de las anteriores.
10	Backtracking es aplicable a problemas de selección y optimización en los que: <ul style="list-style-type: none"> a. El espacio de soluciones es un conjunto infinito. b. El espacio de soluciones es un conjunto finito. c. En cualquiera de los casos
11	En un problema resuelto por backtracking, el conjunto de valores que pueden tomar las componentes de la tupla solución, ha de ser: <ul style="list-style-type: none"> a. Infinito. b. Finito c. Continuo
12	Al aplicar vuelta atrás a la solución de problemas, obtenemos algoritmos con costes computacionales: <ul style="list-style-type: none"> a. Polinómicos. b. Exponentiales. c. Los dos son correctos. Depende del problema.
13	Vuelta atrás se emplea en la resolución de problemas de optimización en los que se pretende encontrar: <ul style="list-style-type: none"> a. Una solución que satisfaga unas restricciones y optimice una cierta función objetivo. b. Todas las soluciones que satisfagan unas restricciones. c. La dos respuestas anteriores son correctas.
14	Backtracking procede a obtener la solución a un problema de optimización empleando la siguiente estrategia: <ul style="list-style-type: none"> a. Genera todas las combinaciones de la solución y selecciona la que optimiza la función objetivo. b. Genera todas las soluciones factibles y selecciona la que optimiza la función objetivo. c. Genera una solución factible empleando un criterio óptimo.

15	Batracking es una técnicas de resolución general de problemas basada en:
	a. La búsqueda sistemática de soluciones. b. La construcción directa de la solución. c. Ninguna de las anteriores
16	Batracking genera las soluciones posibles al problema:
	a. Mediante el recorrido en profundidad del árbol que representa el espacio de soluciones. b. Mediante el recorrido en anchura del árbol que representa el espacio de soluciones c. Ninguna de las anteriores
17	El problema de la mochila, ¿puede solucionarse empleando vuelta atrás?:
	a. Sólo para el caso de la mochila con fraccionamiento b. Sólo para el caso de la mochila sin fraccionamiento c. Se puede aplicar para ambos casos.
18	El problema del viajante de comercio puede resolverse correctamente empleando estos esquemas de programación:
	a. Solo backtracking b. Empleando cualquiera de estos: Voraz y backtracking. c. Sólo programación dinámica.
19	El tiempo de ejecución de un algoritmo de ramificación y poda depende de:
	a. La instancia del problema b. La función de selección de nodos para su expansión c. De ambos
20	Dado un problema resuelto mediante backtracking y mediante ramificación y poda, el coste computacional de la solución por ramificación y poda, en comparación con la de backtracking es:
	a. Igual b. Mayor c. Menor.
21	Cuál de estas afirmaciones es falsa?
	a. Batracking inspecciona todo el espacio de soluciones de un problema mientras que Ramificación y poda no. b. La complejidad en el peor caso de las soluciones Backtracking y ramificación y poda a un mismo problema es la misma. c. Para un mismo problema, ramificación y poda explora siempre un número de nodos menor o igual que backtracking.
EL PROBLEMA DE LA ASIGNACIÓN DE TURNOS.	
	Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.
	Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima
22	El problema de la asignación de turnos tiene solución óptima voraz aplicando la siguiente estrategia:
	a. Seleccionamos los alumnos en orden descendente de preferencia respetando las restricciones de cabida de cada turno. b. Seleccionamos los alumnos en orden ascendente de preferencia respetando las restricciones de cabida de cada turno c. El problema no tiene solución óptima voraz
23	El problema de la asignación de turnos tiene solución óptima empleando:
	a. Batracking b. Voraz c. Ambos
24	El problema de la asignación de turnos tiene solución...
	a. Optima mediante baotracking b. Aproximada (sub-óptima) mediante voraz c. Ambas.
25	Dada la solución recursiva mediante vuelta atrás al problema de la asignación de turnos ¿cuántas nuevas llamadas recursivas genera cada llamada recursiva?
	a. una o dos b. una o ninguna c. ninguna de las anteriores
26	El problema de la asignación de turnos resuelto mediante backtracking tiene una complejidad:
	a. Exponencial b. Polinómica c. Ninguna de la dos