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 pode 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 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 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 sera más rápido.

   (c) Esta estrategia no asegura que se obtenga el camino mas 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 reconsidere 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

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

    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. ¿cual 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() >= pb`
    [(b)] `n.optimistic_b() <= pb`
    (c) `n.pessimistic_b() <= 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.

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

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

   [(b)] No

   (c) Sí, en cualquier caso.

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:

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

   (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 **XXXXXXX**?

```
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(XXXXXXX));
}
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 $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ entonces ...

(a) $\ldots f(n) \in \Theta(g(n))$
[(b)] $\ldots f(n) \in O(g(n))$
(c) $\ldots g(n) \in O(f(n))$

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.

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)] $O(n)$

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 pode el nodo que conduce a la solución óptima.

18. Sea la siguiente relación de recurrencia

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

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

    (a) $g(n) = n^2$

    (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

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

(a) El método voraz

(b) Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

[(c)] Programación dinámica.

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

(a) El valor de una mochila que contiene todos los objetos restantes 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.

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

[(a)] El problema del cambio.

(b) El problema de la asignación de tareas.

(c) La mochila discreta sin restricciones adicionales.

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:

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

Di 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) $O(f) = \{g : \mathbb{N} \to \mathbb{R}^+ | \forall c \in \mathbb{R}, \forall n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \leq cg(n)\}$

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

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

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

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.

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 $\Omega(n^2)$.

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 $O(2^n)$

7. La complejidad temporal del 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.

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

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

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

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

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.optimistic_b() <= pb`

(b) `n.pessimistic_b() <= pb`

[(c)] `n.optimistic_b() >= 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.

18. Si $\lim_{n\to\infty}(f(n)/n^2) = k$, 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) $f(n) \in \Theta(n^2)$

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 ahorra 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) $f \in \Theta(\text{máx}(g_1, g_2))$

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

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

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 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:

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

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

(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:

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

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

    (a) $p > a^k$

    [(b)] $p < a^k$

    (c) $p = a^k$

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

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

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

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

22. Si $\lim_{n\to\infty}(f(n)/n^2) = k$, 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) $f \notin \Theta(\text{máx}(g_1, g_2))$

[(c)] $f \in \Theta(g_1 + g_2)$

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:

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

Di cuál de las siguientes afirmaciones es *cierta*:

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

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

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.

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.