RECOPILACIÓN TESTS EXÁMENES ADA

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 algoritmo 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 las 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 mismo parámetros. Una de las siguientes 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 artistas 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 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 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) € $O(n^2)$, ¿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 pasa 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 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. ¿Que 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, n-r[n]
***c. p, r, n-i**

22. Una de estas tres situaciones no es posible:

a. f(n) Є O(v) y f(n) Є Ω(1)
***b. f(n) Є $\Omega(n^2)$ y f(n) Є O(n)**
c. f(n) Є O(n) y f(n) Є $O(n^2)$

23. Sea A una matriz cuadrada n x 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 problemas es $O(n^2)$.**

24. Cuál de los siguientes algoritmos proveería una cota pesimista para el problema de encontrar el camino más corto entre dos ciudades (se suponen 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 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. ¿Cual de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz (greedy) que sea 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 ellos 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 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 su 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 “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 no $\in O(n^2)$
b. Sólo para valores bajos de n
***c. Es perfectamente posible, ya que $O(n^2) \subset O(n^3)$**

41. 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. pessimistic_b() <= pb

42. ¿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.

43. 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.**

44. 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.

45. El algoritmo de ordenación Quicksort divide el programa 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. $O(n)$**

46. 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.

47. 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.

48. 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. ¿Que 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

49. Sea g(n) = Σa_i*n^i (hasta K con i = 0). Di cuál de las siguientes afirmaciones es falsa:
***a. Las otras dos afirmaciones son ambas falsas.**
b. g(n) є Θ(n^K)
c. g(n) є Ω(n^K)

50. Si lim_n→∞ f(n)/g(n) = 0 entonces…
a. … f(n) є Θ(g(n))
***b. … f(n) є O(g(n))**
c. … f(n) є O(f(n))

51. ¿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.**

52. 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.

53. 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) є O(n), ¿en cuál de estos tres casos nos podemos encontrar?
a. $g(n) = n^2$
b. $g(n) = n$
c. $g(n) = 1$

54. 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.

55. ¿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.

56. ¿Cual 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)

57. ¿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.

58. 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. Θ(2^n)
b. Ω(n) y O(n²)
c. Ω(n) y O(2^n)

59. 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.

60. Cuál de los siguientes criterios proveería una cota optimista para el problema de encontrar el camino más 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.

61. 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) ϵ O(n²) la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda por inserción.
b. Si g(n) ϵ O(n) la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mergesort.
c. Si g(n) ϵ O(1) la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

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

a. O(f) = {g : N -> R^+|$□_c$ ϵ R, $□_{n0}$ ϵ N, $□_n$ >= n_0, f(n) <= cg(n)}

b. **$O(f) = \{g : N \text{->} R^+ | \square_c \in R, \square_{n0} \in N, \square_n >= n_0, g(n) <= cf(n)\}$**
c. $O(f) = \{g : N \text{->} R^+ | \square_c \in R, \square_{n0} \in N, \square_n >= n_0, f(n) <= cg(n)\}$

63. Si f(n) є $O(n^2)$, ¿podemos decir siempre que f(n) є $O(n^3)$?
a. No, ya que n^2 no є $O(n^3)$
b. **Si, ya que n^2 є $O(n^3)$**
c. Sólo para valores bajos de n

64. 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 al a 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. ¿Que 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.**

65. Sea A una matriz cuadrada n x 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 $\Omega(n^2)$

66. 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.**

67. 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.**

68. ¿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.**

69. Si lim_n→∞ $(f(n)/n^2) = k$, y k != 0, ¿cuál de estas tres afirmaciones es falsa?

a. f(n) є $O(n^3)$
b. f(n) є $\Theta(n^3)$
c. f(n) є $\Theta(n^2)$

70. ¿Cuál es la definición correcta de Ω(g)?
a. □(g) = {f : N -> R^+|$\square_c$ ϵ R, $\square_{n0}$ ϵ N, $\square_n$ >= n_0, g(n) >= cf(n)}
b. □(g) = {f : N -> R^+|$\square_c$ ϵ R, $\square_{n0}$ ϵ N, $\square_n$ >= n_0, f(n) >= cg(n)}
c. □(g) = {f : N -> R^+|$\square_c$ ϵ R, $\square_{n0}$ ϵ N, $\square_n$ >= n_0, f(n) >= cg(n)}

71. ¿Cuál es la definición correcta de Ω(f)?
a. □(f) = {g : N -> R^+|$\square_c$ ϵ R, $\square_{n0}$ ϵ N, $\square_n$ >= n_0, g(n) >= cf(n)}
b. □(f) = {g : N -> R^+|$\square_c$ ϵ R, $\square_{n0}$ ϵ N, $\square_n$ >= n_0, f(n) >= cg(n)}
c. □(f) = {g : N -> R^+|$\square_c$ ϵ R, $\square_{n0}$ ϵ N, $\square_n$ >= n_0, f(n) >= cg(n)}

72. 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

73. 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. Θ(log_2(n)) = Θ(log_3(n))
c. $\Theta(\log^2(n)) = \Theta(\log^3(n))$

74. La función γ de un número semientero positivo (un número es semiento 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. Si, pero la complejidad temporal no mejora.

75. ¿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 vuelve a presentar.
c. La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.

76. 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)

77. 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. $n^2 \log(n)$

78. 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.

79. 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 cota 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.

80. ¿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 se la instancia a resolver.
   c. Hay problemas de optimización para los cuales se puede obtener siempre la solución óptima utilizando una estrategia voraz.

81. 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 de las otras dos.
   a. O(n^2) C O(2^log_2(n)) C O(2^n)
   b. (4^log_2(n)) C O(n^2) C O(2^n)
   c. O(2^log_2(n)) C O(n^2) O(n!)

82. Si f є Θ(g1) y f є Θ(g2) entonces
   a. f^2 є Θ(g1*g2)
   b. Las otras dos opciones son ambas ciertas
   c. f є Θ(max(g1, g2))

83. 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) є Θ(1) la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
   b. Si g(n) є Θ(n) la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.
   c. si g(n) є Θ(n) la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.

84. ¿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.

85. 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) є Θ(n^k log_a(n))?
   a. p > a^k
   b. p = a^k
   c. p < a^k

86. Sea A una matriz cuadrada n x 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 esta en $\Omega(n^n)$.
b. La complejidad temporal de la mejor solución posible al problema esta en $O(n \log n)$.
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.

87. ¿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.

88. 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.

89. 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 de las otras dos.

a. $\log(n^3)$ no $\in \Theta(\log_3(n))$
b. $\Theta(\log^2(n)) = \Theta(\log^3(n))$
c. $\Theta(\log_2(n)) = \Theta(\log_3(n))$

90. 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.

91. 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 de las otras dos.
a. $O(n^2)$ C $O(2^{\log_2(n)})$ C $O(2^n)$
b. **$O(2^{\log_2(n)})$ C $O(n^2)$ $O(n!)$**
c. $(4^{\log_2(n)})$ C $O(n^2)$ C $O(2^n)$

92. 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. $p = a^k$

93. 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. $f(n) \in \Theta(n^3)$

94. Si $f \in \Theta(g1)$ y $f \in \Theta(g2)$ entonces
a. $f \in \Theta(g1*g2)$
b. f no $\in \Theta(\max(g1, g2))$
c. **$f \in \Theta(g1+g2)$**

95. 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.

96. ¿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. $\Theta(n)$

97. De las siguientes expresiones, o bien dos 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. **Θ(f) = O(f) □ □(f)**
b. □(f) = Θ(f) □ O(f)
c. O(f) = □(f) □ Θ(f)

98. 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) ϵ Θ(n) la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación quicksort.**
b. Si g(n) ϵ Θ(n) la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación quicksort.
c. si g(n) ϵ Θ(1) la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación mergesort.

99. ¿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.

100. 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.

101. 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.**

102. 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.

103. 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)

104. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
- a. Si, 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.**

105. ¿Qúe 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.

106. Sea la 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) ϵ 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)

107. Si f ϵ Ω(g1) y f ϵ Ω(g2) entonces
- **a. f ϵ Ω(g1 + g2)**

b. f no є Ω(min(g1, g2))
c. f є Ω(g1*g2)

108. 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)

109. 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.

110. 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

111. ¿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.

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

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

113. 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.

114. 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 $\Omega(1)$

115. 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.

116. 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.

117. Dada la siguiente función (donde max(a, b) є $\Theta(1)$)

```
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 $\Omega(n^2)$

118. ¿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. $\Theta(n*m)$

119. 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 la decisiones que se toman no deberían reconsiderarse.

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

a. $T(n) \in \Theta(n)$ y $T(n) \in$ □(n^2)
b. $T(n) \in O(n)$ y $T(n) \in \Theta(n)$
c. $T(n) \in$ □(n) y $T(n) \in \Theta(n^2)$

121. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: pot(x, n) = pot(x, n/2) * pot (x, n/2); ¿Qué esquema resultará 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.

122. 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.

123. 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.

124. 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 de programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?

a. Temporal Θ(n x m) y espacial Θ(n x m)
b. Temporal Θ(n x m) y espacial Θ(min {n, m})
c. Temporal Θ(max{n, m}) y espacial Θ(max{n, m})

125. ¿Qué se deduce d f(n) y g(n) si se cumple lim n->inf (f(n)/g(n)) = k, con k != 0?

a. f(n) ϵ O(g(n)) y g(n) ϵ O(f(n))
b. f(n) ϵ O(g(n)) pero g(n) no ϵ O(f(n))
c. g(n) ϵ O(f(n)) pero f(n) no ϵ O(g(n))

126. 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.

127. 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 divide y vencerás

128. Un tubo de n centimetros 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.

129. 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 de las otras dos.

a. $\Theta(n)$ C $\Theta(n^2)$
b. $n + n \log n \in$ □(n)
c. $O(2^{\log n})$ C $O(n^2)$

130. 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.

131. 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 de ramificación y poda no aporta nada con respecto a vuelta atrás.

132. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿que 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 explorar el nodo que conduce a la solución óptima.
c. Que se podría explorar más nodos de los necesarios

133. 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.

134. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recurrencia 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. $T(n) = T(n - 1) + n$

135. 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. ¿Cual es el orden de complejidad que se obtuvo?

a. $n \log^2 n$
b. n^2
c. $n \log n$

136. Se desea ordenador 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.

137. 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 □(1)
c. La complejidad temporal en el mejor de los casos es $\Theta(n)$

138. 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.

139. ¿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 tiene por qué corresponderse con soluciones factibles.
c. Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

140. 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.

141. Tenemos un conjunto de n enteros positivos y queremos encontrar el sub-conjunto 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.

142. Di cuál de estos resultados de coste temporal asintótico es falso:
a. La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en □(n^2).
b. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en □(n^2).
c. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en O(log n).

143. 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. 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.

144. 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.