1. Un vector de enteros de tamaño n tiene sus elementos estructurados en forma de montículo (heap). Cuál es la complejidad temporal en el peor de los casos de borrar el primer elemento del vector y reconstruirlo posteriormente para que siga manteniendo la estructura de montículo?

a) 0(n).
b) O(log n).
c) O(n log n)

2. El problema del cambio consiste en formar una cantidad dineraria M utilizando el menor número posible de monedas a escoger de entre las disponibles. ¿Qué estrategia resulta ser la más eficiente para garantizar la solución óptima?

a) Un algoritmo voraz.
b) Programación dinámica.
c) Divide y vencerás.

3. En un algoritmo de ramificación y poda. ¿Qué ocurre si coinciden los valores obtenidos por las cotas pesimista y optimista del mismo nodo?

a) Esta situación no puede ocurrir en ningún caso; por lo tanto, una de las cotas está mal calculada (o ambas).
b) Que ese valor es a su vez el mejor valor que se puede obtener con ese nodo.
c) Que es un nodo hoja, esta situación sólo es posible en los nodos hoja.

4. En un algoritmo de ramificación y poda. ¿Qué ocurre si coinciden los valores obtenidos por las cotas pesimista y optimista del mismo nodo?

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

5. Sólo una de las tres afirmaciones siguientes es cierta. Cuál?

a) **Mientras que el algoritmo de Prim va construyendo un único árbol de recubrimiento añadiendo vértices uno a uno, el algoritmo de Kruskal va construyendo un bosque que va uniendo añadiendo aristas, hasta obtener un único árbol de recubrimiento.**
b) Mientras que el algoritmo de Prim va construyendo un único árbol de recubrimiento añadiendo aristas una a una, el algoritmo de Kruskal va construyendo un bosque que va uniendo añadiendo vértices, hasta obtener un único árbol de recubrimiento.
c) Los algoritmos de Prime y de Kruskal mantienen en todo momento un único árbol de recubrimiento, que crece añadiendo aristas o vértices.

6. Cuál es la complejidad espacial del algoritmo Quicksort?

a) 0(n).
b) O(n log n).
c) O(1)

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

T(n) = { 1 n <= 1
T(n/2) + g(n) n> 1

De las siguientes afirmaciones, o bien dos son ciertas y una es falsa, o bien dos son falsas y una es cierta. Marca la opción que en este sentido es diferentes a las demás.

a) Si g(n) ϵ O (n) la relación de recurrencia representa la complejidad temporal en el peor caso del algoritmo de búsqueda del k-ésimo elemento más pequeño de un vector (estudiado en clase).
b) Si g(n) ϵ O (1) la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
c) Si g(n) ϵ O (n^2) la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria

8. Se pretende borrar todos los elementos de un vector cuyo valor es un número par. Si el tamaño del vector n, ¿con qué coste temporal asintótico se podría realizar esa operación?

a) 0(n).
b) O(n log n).
c) O(1)

9. Por qué muchos algoritmos voraces presentan complejidades temporales en O(n log n)?

a) **Porque primero ordenan de alguna manera los elementos y porque una vez ordenados la complejidad temporal del proceso de selección de los elementos que se incorporarán a la solución está en O(n log n).**
b) Porque primero ordenan de alguna manera los elementos y porque una vez ordenados la complejidad temporal del proceso de selección de los elementos que se incorporarán a la solución está en O(n log n).
c) Porque el proceso de selección de los elementos que se incorporarán en la solución es siempre O(n log n).

10. Una de las siguientes afirmaciones es falta. Cuál?

a) **Si f ϵ O(g) y g ϵ O(f), entonces f = g**
b) Si f ϵ O(g) y g ϵ O(f), entonces O(f) = O(g)
c) Si f ϵ O(g) y g ϵ O(h), entonces f ϵ O(h)

11. Uno de estos tres algoritmos no resuelve el mismo problema que los otros dos. Cuál?

a) **El algoritmo de Floyd y Warshall.**
b) El algoritmo de Prim.
c) El algoritmo de Kruskal.

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

```
int f(int n) {
int count = 0;
 for (int i = n; i > 0; i /= 2)
    for(int j = 0; j < 2 * i; j++)
       count += 1;
return count;
}
```

a) **O(n)**
b) O(n log n)
c) O(n^2)

13. En los algoritmos de Ramificación y poda …

a) Una cota pesimista es necesariamente un valor insuperable, de no ser así, se podría podar el nodo que conduce a la solución óptima.
b) **Una cota pesimista es necesariamente un valor alcanzable, de no ser así, no está garantizado que se encuentre la solución óptima.**
c) Una cota optimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo

14. Puede ser que una solución recursiva con memoización a un problema de optimización realice menos evaluaciones de la función que computa el valor de una solución parcial que una basada en programación dinámica iterativa y por lo tanto acabe siendo más rápida?

a) Sí; de hecho, esto pasa con el problema de la mochila discreta con pesos enteros.

b) No: las dos soluciones tienen que realizar el mismo número de evaluaciones de la función que computa el valor de una solución parcial, y la solución recursiva es más lenta porque tiene que gestionar las llamadas recursivas (argumentos, reserva de espacio temporal, etc,).

c) No: las soluciones recursivas realizan más evaluaciones de la función que computa el valor de una solución parcial que las iterativas correspondientes.

15. Cuando se usan cotas pesimistas para hacer podas en algoritmos de optimización basados en búsqueda y enumeración (por ejemplo, vuelta atrás o ramificación y poda) …

a) …se pueden obtener sin visitar necesariamente las hojas del árbol de búsqueda.

b) ...es posible que no encontramos la solución óptima.

c) …siempre se obtienen cuando se visitan las hojas del árbol de búsqueda.

16. ¿Qué desventaja presenta la solución de ramificación y poda frente a la solución de vuelta atrás aplicadas al problema de la mochila?

a) La complejidad espacial: En el caso más desfavorable, el coste espacial asintótico de un algoritmo de ramificación y poda es exponencial, mientras que en vuelta atrás no lo es.

b) Ninguna de las otras dos opciones es cierta, tanto la complejidad espacial como la temporal de ambas técnicas es similar en el caso más desfavorable.

c) La complejidad temporal: En ramificación y poda es más probable que una instancia pertenezca al caso peor.

17. Qué aporta la técnica de ramificación y poda frente a la de vuelta atrás?

a) La posibilidad de combinar el uso de cotas pesimistas y optimistas para cualquier nodo ya sea completado o sin completar. En vuelta atrás esto no se puede hacer.

b) La posibilidad de analizar diferentes estrategias para seleccionar el siguiente nodo a expandir.

c) Eficiencia. Los algoritmos de ramificación y poda son más eficientes que los de vuelta atrás.

18. Se puede hacer que la solución de programación dinámica iterativa al problema de la mochila discreta con pesos enteros tenga una complejidad espacial que no dependa del número de objetos?

a) **No: el cálculo de la fila k de la matriz (correspondiente a haber considerado k objetos) necesitamos todas las k - 1 filas anteriores.**
b) Sí, porque la solución es realmente un algoritmo voraz con complejidad espacial constante.
c) Sí, porque para calcular la fila k de la matriz (correspondiente a haber considerado k objetos) sólo necesitamos la fila k-1 (correspondiente a haber considerado k - 1 objetos).

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

a) O(2^n)
b) O(n!)
c) O(n^2)

20. La solución de programación dinámica iterativa del problema de la mochila discreta…

a) **tiene la restricción de que los valores tienen que ser enteros positivos.**
b) calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
c) tiene la restricción de que los valores tienen que ser enteros positivos.

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

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

22 Supongamos el algoritmo de ordenacion Mergesort modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres Posteriormente se combinan las soluciones parciales. Cual seria la complejidad temporal del nuevo algoritmo?

n log(n)

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

a. El problema de la asignación de tareas.
b. La mochila discreta sin restricciones adicionales.
c. El problema del cambio (devolver una cantidad de dinero con el mínimo número de modas).

24 ¿cuál afirmación es falsa?
a: Los algoritmos iterativos de programación dinámica utilizan memorización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar
b: La solución de programación dinámica iterativa al problema de la mochila discreta realiza cálculos innecesarios.
c: La memorización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.

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

a. Para actualizar el valor de la mejor solución hasta el momento.
b. Para descartar el nodo si no es prometedor.
c. Para obtener una cota optimista más precisa.

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. Coste exacto(log^2(n)) = Coste exacto(log^3(n))
b. log(n^3) no pertenece Coste exacto(log3(n))
c. Coste exacto(log2(n)) = Coste exacto(log3(n))

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

28. 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.

29. ¿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. Coste exacto(log n)

b. Coste exacto(log^2 n)

c. Coste exacto(n)

30. Sea A una matriz cuadrada n x n. Se trata de buscar una permutacion de las columnas tal que la suma de los elementos de la diagonal de la matriz sea minima. Indicad cuál de las siguientes afirmaciones es **correcta**.

c. Si se construye una solución al problema basada en el esquema de ramifación y poda, una buena elección de cotas optimistas y pesimistas podría evitar la exploración de todas las permutaciones posibles.

31. ¿Para qué sirven las cota pesimistas en ramificación y poda?

a) Para tener la certeza de que la cota optimista está bien calculada.

b) Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.

c) Para descartar nodos basándose en el beneficio esperado.

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

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

a. cuadrática
b. lineal
c. cúbica

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

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

34 si T(n) € O(n^2) en cuál de estos casos nos podemos encontrar?
a: g(n)=n
b g(n)=n^2
c g(n)=n log(n)

35:

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

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

(b) $f^2 \in \Theta(g_1 \cdot g_2)$

c: Las otras dos opciones son ambas ciertas.

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

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